Základy PHP; jednoduché skriptování

Co se v modulu dozvíte?

- Co je a k čemu slouží PHP
- Jak s PHP začít pracovat
- Jak implementovat PHP kód do kódu HTML stránky
- Jednoduché ukázky příkazů a zdrojového kódu
- Hlavní informační zdroje

K čemu lze PHP využít? V PHP je možno vytvořit nejen jednoduché skripty, ale celé webové aplikace. Jelikož se jedná o server-side skriptovací jazyk, hodí se všude tam, kde je potřeba skrýt zdrojový kód aplikace před koncovým uživatelem. PHP kód je zpracováván na serveru a ten poté zasílá webovému prohlížeči pouze jeho výsledek. Dále se PHP hodí pro komunikaci a manipulaci s databází, která je umístěna taktéž na serveru.

Příklady využití PHP

- Generování dynamického obsahu na definovaném místě stránky
- Zpracování dat z webových formulářů
  - validace
  - vložení obsahu formuláře do databáze
  - odeslání na e-mail,...
- Dotazování a manipulace s daty v databázi
  - kniha návštěv
  - to-do list
  - katalog knih,...
- Vytvoření chráněné sekce stránek jen pro oprávněné uživatele
  - stránka chráněná heslem
  - administrační část webu
- Vytváření, úprava, čtení a mazání souborů na serveru
  - upload obrázků do fotogalerie
  - generování souborů dle parametrů (XML, CSV,...)
- Šifrování dat a mnoho dalšího

Jak s PHP začít pracovat?

Pro základní práci s PHP postačí běžný editor s podporou zvýraznění PHP syntaxe (zdrojového kódu), webový prohlížeč a nainstalovaný webový server s podporou PHP.

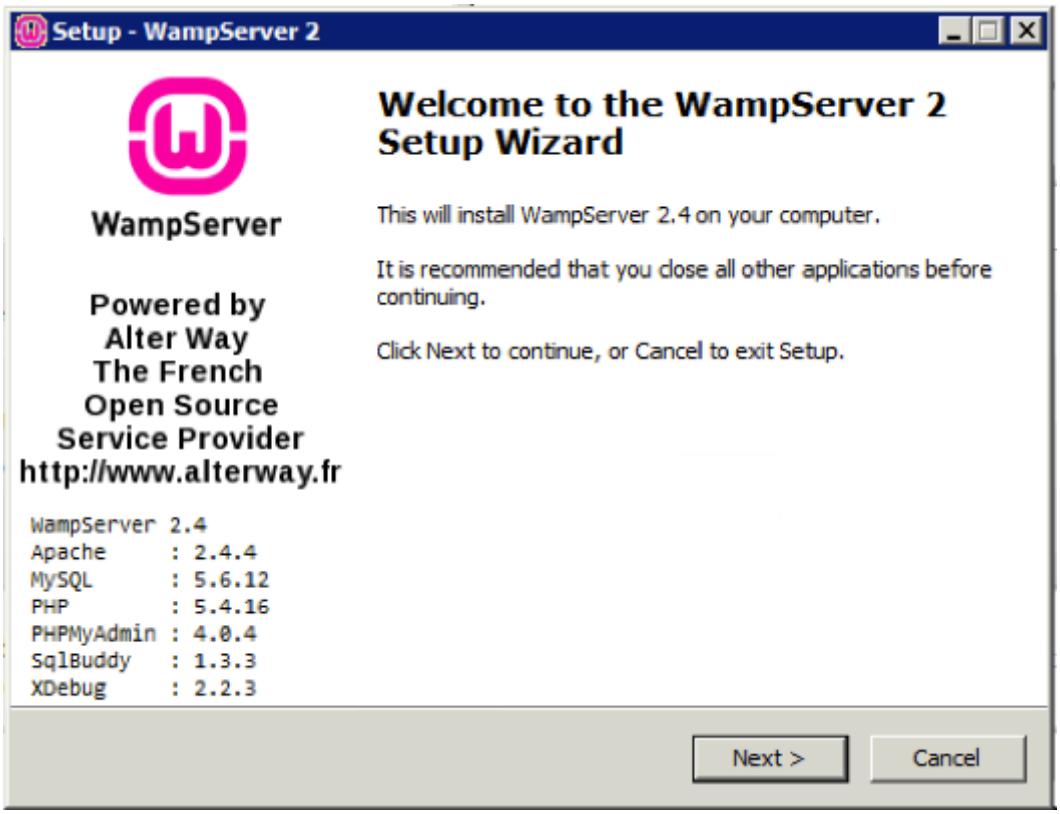
U většiny poskytovatelů hostingu pro webové stránky již v základu poskytuje podporu pro PHP. Není však od věci tuto skutečnost zkontrolovat v popisu služby či se dotázat na technickou podporu

poskytovatele. Pro lokální vývoj a testování webových stránek, které obsahují PHP kód, je nutné nejprve ve svém počítači nainstalovat webový server.

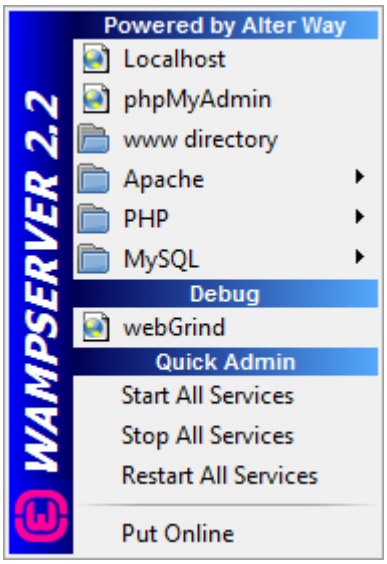
Instalace webového serveru na lokální počítač

Samotná instalace je díky komplexním instalačním balíčkům jednoduchá a zvládne ji i běžný uživatel. Jedním z balíčků, který lze pro instalaci webového serveru na lokálním počítači použít, je tzv. WampServer.

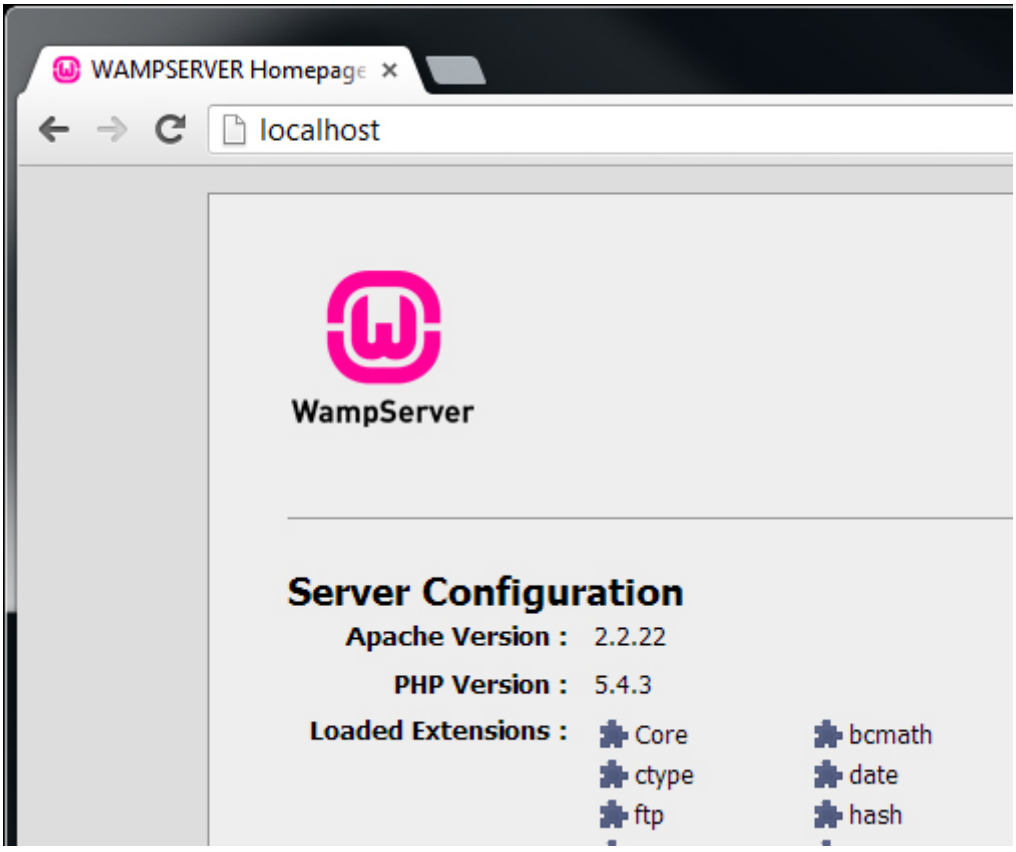
Dle instalačních instrukcí na http://www.wampserver.com/en/ stáhněte a nainstalujte balíček. Ten obsahuje vše potřebné ke zprovoznění PHP na vlastním počítači (server Apache, modul PHP pro Apache, databázi MySQL a webové rozhraní pro správu databáze PhpMyAdmin).



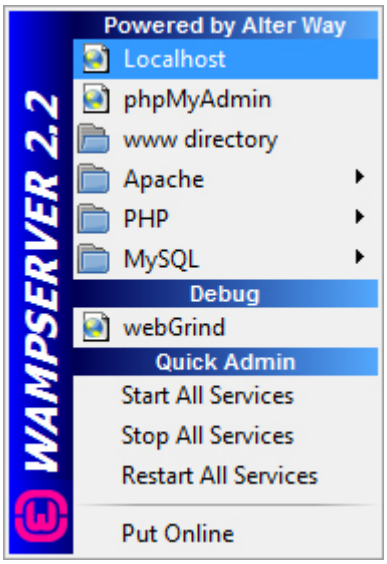
Pokud jste provedli instalaci správně a spustíte WampServer, najdete v systémové liště (Taskbar) Windows ikonu WampServeru. Správnou funkčnost spuštění služeb serveru indikuje zelená barva ikony. Po kliknutí na tuto ikonu se zobrazí menu, kterým lze ovládat nastavení serveru a spravovat ho. Pokud je ikona červená nebo oranžová, nepodařilo se spustit všechny služby serveru úspěšně. Příčinnou může být kolize s obsazenými komunikačními porty jiných aplikací – např. Skype. Ukončete tyto aplikace a spusťte WampServer znovu.



Pro otestování funkčnosti serveru otevřete webový prohlížeč a do adresní řádky napište `localhost`. Měla by se zobrazit úvodní stránka WampServeru.



K úvodní stránce serveru lze přistupovat i prostřednictvím menu WampServeru po kliknutí na položku Localhost.



Založení vlastního projektu

Všechny projekty jsou tříděny v adresářích, jejichž seznam je možné vidět na úvodní stránce pod nadpisem Your Projects . Nový projekt vytvoříme následovně:

- Kliknutím na ikonu WampServeru v systémové liště a zvolením položky WWW directory, otevřete složku webových projektů (tento adresář má při spuštění WampServeru adresu localhost).
- Vytvořte nový adresář a pojmenujte ho výstižným názvem, např. zaklady-php . Nepoužívejte diakritiku a speciální znaky.
- Otevřením (nebo obnovením) úvodní stránky WampServeru ve webovém prohlížeči uvidíte svůj nový adresář v seznamu projektů.
- Do tohoto adresáře budeme vkládat všechny soubory a složky, které se týkají daného projektu.

Implementace PHP kódu do HTML kódu stránky

Do HTML kódu lze PHP kód vkládat na libovolné místo. Je nutné však dávat pozor na správnou syntaxi a pořadí HTML prvků (tagů). PHP kód je do stránky vložen pomocí značek <?php pro začátek PHP kódu a ?> pro konec PHP kódu. Vložený PHP kód do stránky tedy může vypadat následovně:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Vložený PHP kód do HTML stránky</title>
  </head>

  <body>
    <h1>Nadpis</h1>
    <p>Běžný HTML odstavec</p>

    <?php

      // toto je komentář - níže je uveden PHP kód
      // každý řádek uceleného příkazu PHP kódu musí končit středníkem
      echo '<p>Odstavec generovaný pomocí PHP</p>';

      /* takto je označen komentář,
         který je na více řádků */

    ?>

    <p>Další běžný HTML odstavec</p>
  </body>
</html>
```

Pro správnou interpretaci PHP kódu musí mít soubor koncovku .php, nikoliv .html a musí být umístěn v projektovém adresáři. Poté stačí najít příslušný projektový adresář pomocí úvodní stránky WampServeru a zobrazit soubor s příkladem.

Komentáře v PHP kódu jsou označeny lomítky nebo lomítkem a hvězdičkou pro víceřádkový komentář.

Jednoduché ukázky základních příkazů

Použití proměnných

Proměnné slouží k uložení hodnoty a jejího pozdějšího použití pro výpis nebo další operace. Proměnné jsou v PHP označeny znakem $ a následuje název proměnné. Ten musí začínat písmenem nebo podtržítkem. V názvu jsou povoleny pouze alfa-numerické znaky (a-z, A-Z, 0-9 nebo podtržítko).

```php
<?php

  // vložení hodnoty do proměnné
  $nazev_vyrobku = 'Počítač';
  $cena = 1000;

  // operace s proměnnými - viz. PHP manuál - operátory
  $sazbaDPH = 21;
  $cena_sDPH = $cena + $cena * ($sazbaDPH / 100);

  /*
  - proměnné lze přiřadit celý řetězec, který je složen např. z kombinace jiných proměnných a textu
  - text musí být vždy v uvozovkách
  - jednotlivé části textu a proměnné lze slepovat do jednoho řetězce pomocí tečky
  */
  $vystup = $nazev_vyrobku.' má cenu '.$cena.' bez DPH, což je '.$cena_sDPH.' včetně DPH (DPH '.$sa
  echo $vystup;

?>
```

Pro přiřazování hodnot a výpočty se používají tzv. přiřazovací a aritmetické operátory.

http://www.w3schools.com/php/php_operators.asp

Podmínky

Využití podmínek je velmi široké a nelze se bez nich téměř obejít. Jde ve své podstatě o výrok, který v případě své platnosti vykoná určitou část kódu. Podmínky se používají k rozhodování a větvení programu.

V PHP je podmínka prezentována klíčovým slovem `if` a `else`, což odpovídá i anglickému významu. Různou kombinací podmínek lze dosáhnout specifického chování programu.

V podmínkách se používají tzv. porovnávací a logické operátory. Pomocí nich lze vytvořit výrok, který má podmínka splňovat.

http://www.w3schools.com/php/php_operators.asp

V případě, že jedna proměnná může nabývat více hodnot, pro které je třeba vykonat specifickou část kódu, je vhodné použít místo použití několika podmínek IF příkaz SWITCH. Ten lze rozdělit na libovolné množství případů (podmínek), které pro danou proměnnou platí.

```php
<?php

  // vložení hodnoty do proměnné
  $nazev_vyrobku_1 = 'Počítač';
  $cena_1 = 1000;
  $barva_1 = "černá";
```

```php

    $nazev_vyrobku_2 = 'Tablet';
    $cena_2 = 500;
    $barva_2 = "bílá";

    // výpis proměnných
    echo '<h1>'.$nazev_vyrobku_1.'</h1>';
    echo '<p>Cena: '.$cena_1.'</p>';
    echo '<p>Barva: '.$barva_1.'</p>';

    echo '<h1>'.$nazev_vyrobku_2.'</h1>';
    echo '<p>Cena: '.$cena_2.'</p>';
    echo '<p>Barva: '.$barva_2.'</p>';


    echo '<h2>Podmínky</h2>';


    // zápis jednoduché podmínky IF
    if ($cena_1 > 0) {
       // kód, který se v případě splnění podmínky vykoná
       echo '<p>Cena zboží s názvem "'.$nazev_vyrobku_1.'" je větší než nula.</p>';
    }



    // zápis podmínky s IF ELSE
    if ($barva_1 == "černá") {
       // kód, který se v případě splnění podmínky vykoná
       echo '<p>Barva zboží s názvem "'.$nazev_vyrobku_1.'" je černá.</p>';
    }
    else {
       echo '<p>Barva zboží s názvem "'.$nazev_vyrobku_1.'" není černá.</p>';
    }



    // zápis podmínky s IF ELSEIF
    if ($cena_1 > $cena_2) {
       // kód, který se v případě splnění podmínky vykoná
       echo '<p>Cena zboží s názvem "'.$nazev_vyrobku_1.'" je větší než u výrobku '.$nazev_vyrobku_2.
    }
    else if ($cena_1 == $cena_2) {
       echo '<p>Ceny výrobků se rovnají.</p>';
    }
    else {
       echo '<p>Cena zboží s názvem "'.$nazev_vyrobku_1.'" je menší než u výrobku '.$nazev_vyrobku_2.
    }


    // pro rozlišení více stavů proměnné je možno použít příkaz SWITCH
    switch ($barva_2) {
      // klíčovým slovem CASE je vyjádřen případ, který má pro danou část kódu platit
        case "černá":
         // kód, který se vykoná platí-li, že proměnná obsahuje řetězec "černá"
           echo 'Barva pro '.$nazev_vyrobku_2.' je černá.</p>';
         // konec kódu pro jednotlivý případ musí být ukončen příkazem break; , který ukončí další po
           break;
      case "bílá":
           echo 'Barva pro '.$nazev_vyrobku_2.' je bílá.</p>';
           break;
      case "modrá":
           echo 'Barva pro '.$nazev_vyrobku_2.' je modrá.</p>';
           break;
      case "červená":
           echo 'Barva pro '.$nazev_vyrobku_2.' je červená.</p>';
           break;
      case "růžová":
           echo 'Barva pro '.$nazev_vyrobku_2.' je růžová</p>';
           break;
    }

?>
```

Cykly

Často lze narazit na problém, kdy je potřeba, aby se část kódu vykonávala několikrát po sobě, dokud není splněna určitá podmínka nebo nenastane specifická situace. Rovněž lze cykly využít pro procházení nebo plnění proměnných, speciálně pak tzv. polí, o kterých ještě bude dále řeč. Všechny cykly lze vnořovat a různě kombinovat, je však nutné dávat pozor na správné ukončení jednotlivých cyklů a vhodnou volbu podmínek.

WHILE

Prvním z typů je tzv. cyklus WHILE, který zajišťuje opakování části kódu, dokud není splněna určitá podmínka. Výrok podmínky je zapisován stejně, jako je tomu u příkazu IF. Cyklus WHILE se používá často také při výpisu dat z databáze.

```php
<?php

  // vložení hodnoty do proměnné
  $pocet_stran = 10;
  $strana = 1;

  echo '<h1>Seznam stran</h1>';
  echo '<ul>';

  // ukázka cyklu WHILE, který probíhá, pokud je hodnota proměnné $strana menší nebo rovna hodnotě
  while ($strana <= $pocet_stran) {
    // část kódu, která se má opakovat
    echo '<li><a href="strana_'.$strana.'.php">Strana '.$strana.'</a></li>';
    // operátor ++ přičte k proměnné jedničku, což je zkrácená verze kódu: $strana = $strana + 1;
    $strana++;
  }

  echo '</ul>';




  // vložení hodnoty do proměnné
  $pocet_stran = 10;
  $strana = 1;

  echo '<h1>Druhý seznam stran</h1>';
  echo '<ul>';

  // ukázka cyklu DO WHILE, který je spuštěn poprvé vždy a další průběh cyklu je závislý na platnos
  do {
    // část kódu, která se má opakovat
    echo '<li><a href="strana_'.$strana.'.php">Strana '.$strana.'</a></li>';
    // operátor ++ přičte k proměnné jedničku, což je zkrácená verze kódu: $strana = $strana + 1;
    $strana++;
  }
  // cyklus se opakuje do té doby, dokud platí, že se nerovnají hodnoty proměnných $strana a $pocet
  while ($strana != $pocet_stran);

  echo '</ul>';




  // vložení hodnoty do proměnné
  $pocet_radku = 4;
  $pocet_sloupcu = 3;

  // inicializace proměných s počáteční hodnotou
  $aktualni_radek = 1;
  $aktualni_sloupec = 1;

  echo '<h1>Tabulka generovaná PHP</h1>';
  echo '<table>';

  // cyklus pro generování řádků tabulky do stanoveného počtu
  while ($aktualni_radek <= $pocet_radku) {
    // ** kód, který je opakován prvním cyklem
    echo '<tr>';

    // pro každý řádek je nutné nastavit proměnnou $aktualni_sloupec na počáteční hodnotu 1
    $aktualni_sloupec = 1;
```

```
63
64      // vnořený cyklus, který pro každý řádek vygeneruje patřičný počet sloupců
65      while ($aktualni_sloupec <= $pocet_sloupcu) {
66        // ++ kód, který je opakován druhým cyklem
67        echo '<td>Buňka ['.$aktualni_radek.','.$aktualni_sloupec.']</td>';
68        $aktualni_sloupec++;
69        // ++ konec kódu, který je opakován druhým cyklem
70      }
71
72      $aktualni_radek++;
73      echo '</tr>';
74      // ** konec kódu, který je opakován prvním cyklem
75    }
76
77    echo '</table>';
78
79
80
81  ?>
```

FOR

Dalším typem cyklu je cyklus FOR. Jeho hlavní předností je možnost určit počet opakování cyklu. Určitou část kódu lze tedy spustit přesně tolikrát, kolikrát je potřeba. Cyklus FOR lze ve většině případů nahradit cyklem WHILE, ale použití cyklu FOR je v některých případech více kompaktní. Cykly FOR lze také vnořovat, případně kombinovat s cykly WHILE.

```
1   <?php
2
3
4     echo '<h1>Letopočty</h1>';
5     echo '<ul>';
6
7     /*
8     ukázka cyklu FOR
9     (první část tvoří přiřazení počátečního stavu; druhá část je hodnota, po kterou má cyklus běžet;
10    */
11    for ($rok = 2010; $rok <= 2020; $rok++) {
12        echo '<li>'.$rok.'</li>';
13    }
14
15    echo '</ul>';
16  ?>
```

FOREACH

Cyklus FOREACH je používán především pro výpis polí. Jeho použití je ukázáno v podkapitole Pole.

POLE

Pole jsou speciálním druhem proměnných, do kterých lze uložit více než jednu hodnotu. Lze si je představit např. jako box, který má své jméno a obsahuje libovolné množství šuplíků, jež jsou označeny také vlastním jménem nebo číselným indexem. Často sdružují větší množství hodnot, které mají nějakou společnou vlastnost, a nevíme přesně, kolik těchto hodnot budeme chtít použít nebo se jejich počet bude dynamicky měnit.

Proměnnou typu pole lze v kódu poznat podle toho, že je za jejím názvem (což je název zmíněného „boxu“) hodnota uzavřená v hranatých závorkách (názvy jednotlivých „šuplíků“. Inicializaci pole lze provést funkcí array(); nebo pouhým přiřazením hodnoty do proměnné s hranatými závorkami, viz. příklad.

```
1   <?php
2
3     // inicializace pole pomocí funkce array();
4     // při re-inicializaci na již existující pole se všechny hodnoty a indexy smažou
```

```php
$moje_pole = array();


/* ********************************************************************** */

/*
vložení hodnot do indexovaného pole; indexování začíná od nuly!

$znacky_pocitacu[0] = "HP";
$znacky_pocitacu[1] = "Asus";
$znacky_pocitacu[2] = "Dell";
...

indexy mohou být načítány automaticky pomocí zápisu:
$znacky_pocitacu[] = "HP";
$znacky_pocitacu[] = "Asus";
$znacky_pocitacu[] = "Dell";
...

hodnoty lze vložit i při inicializaci pole zápisem:
*/
$znacky_pocitacu = array("HP", "Asus", "Dell", "Acer", "Lenovo");

echo '<h1>Výpis jednoho prvku pole</h1>';
// vypsání prvku pole s indexem 2, vypsána bude hodnota: Dell
echo $znacky_pocitacu[2];


/* ********************************************************************** */


// použití cyklu pro výpis pole

echo '<h1>Značky počítačů</h1>';

// funkce count(); spočítá počet prvků v poli - v našem případě bude mít hodnotu 5.
$pocet_prvku_pole = count($znacky_pocitacu);

// cyklem FOR lze procházet hodnoty prvků v poli
for ($i = 0; $i < $pocet_prvku_pole; $i++) {
    echo '<p>'.$znacky_pocitacu[$i].'</p>';
}


/* ********************************************************************** */


// použití cyklu pro načtení hodnot do pole

// inicializace pole
$prestupne_roky = array();

$mnozstvi_letopoctu = 8;
$letopocet = 2000;

echo '<h1>Přestupné roky</h1>';

// cyklem FOR lze plnit pole hodnotami
for ($i = 0; $i < $mnozstvi_letopoctu; $i++) {
    $prestupne_roky[] = $letopocet;
    $letopocet = $letopocet + 4;
}

// výpis pole cyklem WHILE
$i = 0;
while ($i < $mnozstvi_letopoctu) {
    echo '<p>'.$prestupne_roky[$i].'</p>';
    $i++;
}

/* ********************************************************************** */


// použití příkazu continue; v cyklech

echo '<h1>Značky počítačů - vynechání prvku pole</h1>';

$pocet_prvku_pole = count($znacky_pocitacu);

for ($i = 0; $i < $pocet_prvku_pole; $i++) {
```

```php
        /*
        - podmínka, která má být splněna pro příkaz continue;
        - v tomto případě se má hodnota prvku pole rovnat řetězci Dell
        */
        if ($znacky_pocitacu[$i] == "Dell") {
          // příkaz continue; se provede při splnění podmínky a přeruší aktuální běh cyklu; cyklus v
          continue;
        }
        echo '<p>'.$znacky_pocitacu[$i].'</p>';
    }


    // použití příkazu break; v cyklech

    echo '<h1>Značky počítačů - vyselektování prvku z pole</h1>';

    $pocet_prvku_pole = count($znacky_pocitacu);

    for ($i = 0; $i < $pocet_prvku_pole; $i++) {
        /*
        - podmínka, která má být splněna
        - v tomto případě se má index pole rovnat hodnotě: 1 (druhý prvek pole, jelikož index začíná
        */
        if ($i == 1) {
          echo '<p>'.$znacky_pocitacu[$i].'</p>';
          // příkaz break; provede přerušení běhu cyklu, bez pokračování následujícím během;
          break;
        }

    }
?>
```

Kromě indexovaných polí existují také tzv. pole asociativní. Jejich funkce je podobná, jako u polí indexovaných, avšak místo číselného indexu je použit textový řetězec – klíč, což je vlastně pojmenovaný index. Asociativní pole jsou často využívána pro výpis záznamů z databáze, viz. podkapitola Práce s databází.

```php
<?php

  /*
  - vložení klíčů a hodnot do asociativního pole
  - jednotlivé záznamy v poli jsou odděleny čárkou
  - zápis odpovídá také zápisu kódu:

    $vlastnosti_vyrobku["hmotnost"] = "0.5";
    $vlastnosti_vyrobku["barva"] = "černá";
    $vlastnosti_vyrobku["ean"] = "8523214789631";
    $vlastnosti_vyrobku["materiál"] = "plast";
  */

  $vlastnosti_vyrobku = array("hmotnost" => "0.5", "barva" => "černá", "ean" => "8523214789631", "m

  echo '<h1>Vlastnosti výrobku</h1>';

  /*
  - pro výpis asociativního pole, jeho klíčů a hodnot, je vhodné použít cyklus FOREACH
  - proměnná $klic bude v každém běhu cyklu obsahovat název klíče a proměnná $hodnota konkrétní hod
  - cyklus FOREACH probíhá tolikrát, kolik je prvků v poli
  */
  foreach ($vlastnosti_vyrobku as $klic => $hodnota) {
    echo '<p>'.$klic.' - '.$hodnota.'</p>';
  }
?>
```

Pro práci s poli existuje mnoho příkazů a funkcí. Pole tak lze např. třídit, řadit, vyhledávat určité klíče nebo hodnoty, atd. Mezi nejdůležitější funkce patří:

- `in_array();` - pro vyhledávání hodnot v poli

- `implode();` - pro vygenerování jednoduchého řetězce z prvků pole, přičemž jsou jednotlivé hodnoty odděleny vybraným znakem/znaky.
- `explode();` - opačný příkaz jako implode – z jednoduchého řetězce vygeneruje pole, přičemž jako oddělovač jednotlivých záznamů pole je použit vybraný znak/znaky.
- `unset();` - pro smazání záznamu v poli

```php
<?php

  // vložení klíčů a hodnot do asociativního pole
    $vlastnosti_vyrobku["hmotnost"] = "0.5";
    $vlastnosti_vyrobku["barva"] = "černá";
    $vlastnosti_vyrobku["ean"] = "8523214789631";
    $vlastnosti_vyrobku["materiál"] = "plast";


  echo '<h1>Příklad funkce in_array();</h1>';

  /*
  - funkce in_array(); je používána většinou ve spojení s podmínkou
  - argumentem je vyhledávaný řetězec a proměnná typu pole
  */
  if (in_array("černá", $vlastnosti_vyrobku)) {
    echo '<p>Výrobek existuje v černé barvě</p>';
  }
  if (!in_array("dřevo", $vlastnosti_vyrobku)) {
    echo '<p>Výrobek není dřevěný</p>';
  }



  echo '<h1>Příklad funkce implode();</h1>';

  // funkce implode spojí všechny hodnoty pole do řetězce a oddělí je v našem případě čárkou
  $seznam_hodnot_pole = implode(", ", $vlastnosti_vyrobku);
  echo '<p>'.$seznam_hodnot_pole.'</p>';



  echo '<h1>Příklad funkce explode();</h1>';

  // funkce explode rozdělí řetězec na části podle určitého znaku (v našem případě znak |) a hodnot
  $retezec = "strana 1|strana 2|strana 3|strana 4";
  $pole = explode("|", $retezec);

  // výpis hodnot pole
  foreach ($pole as $hodnota) {
    echo '<p>'.$hodnota.'</p>';
  }



  echo '<h1>Příklad funkce unset();</h1>';

  // funkce unset() smaže prvek pole; v našem případě na třetí pozici a hodnotou "strana 3" (indexo
  unset($pole[2]);

  // výpis hodnot pole
  foreach ($pole as $hodnota) {
    echo '<p>'.$hodnota.'</p>';
  }


?>
```

Přehled dalších funkcí pro práci s poli najdete v PHP Manuálu - http://php.net/manual/en/ref.array.php

Funkce

Funkce v PHP je možné využít stejně, jako v jiných programovacích jazycích. Funkce je vlastně část kódu, která má vykonávat určitou specifickou činnost. Do funkcí se sdružuje kód, který je často používán a plní jednoznačný obecný úkol. Funkce tak omezují duplicitu zdrojového kódu na více místech programu. V PHP je možno najít funkce předdefinované – jsou základní součástí PHP balíčku a funkce uživatelské – vlastní funkce napsané programátorem. Předdefinovaných funkcí je velké množství a celkový jejich přehled lze nalézt v PHP Manuálu: http://php.net/quickref.php http://php.net/manual/en/funcref.php W3Schools - PHP References: http://www.w3schools.com/php/default.asp

Vlastní funkce je vhodné pojmenovávat tak, aby bylo zřejmé, jakou činnost zhruba provádějí. Usnadníte tím orientaci v kódu nejen sobě, ale také dalším programátorům.

```php
<?php


  /*
  - funkce je definována klíčovým slovem "function", za ním následuje název funkce a její parametry
  - parametry funkce jsou proměnné, jež jsou dostupné pouze v dané funkci
  */
  function soucet($prvni_cislo, $druhe_cislo) {
    $vysledek = $prvni_cislo + $druhe_cislo;

    // funkce má vrátit hodnotu proměnné $vysledek
    return $vysledek;
  }

  echo '<h1>Jednoduchá funkce</h2>';
  // výpis s voláním vlastní funkce soucet(), v závorce jsou uvedeny parametry, které se automatick
  echo '<p>Součet je: '.soucet(8,2).'</p>';




  echo '<h1>Použití funkce date();</h2>';

  // příklad použití předdefinované funkce date(), která v našem případě vrátí aktuální letopočet -
  echo 'Je právě rok '.date("Y");


?>
```

Hlavní informační zdroje

- PHP Manual - http://www.php.net/manual/en/
- W3schools - http://www.w3schools.com/php/
- http://www.codecademy.com/tracks/php

Tyto materiály vznikly v rámci projektu CZ.2.17/3.1.00/34129
Rozvoj oboru Multimédia v ekonomické praxi pro lepší uplatnění absolventů v praxi

Evropský sociální fond - Praha & EU: Investujeme do vaší budoucnosti