PHP

Ing:DRGO Pavel,27.júl 2016

OBSAH

1. díl - Úvod do PHP a webových aplikací

Webové stránky

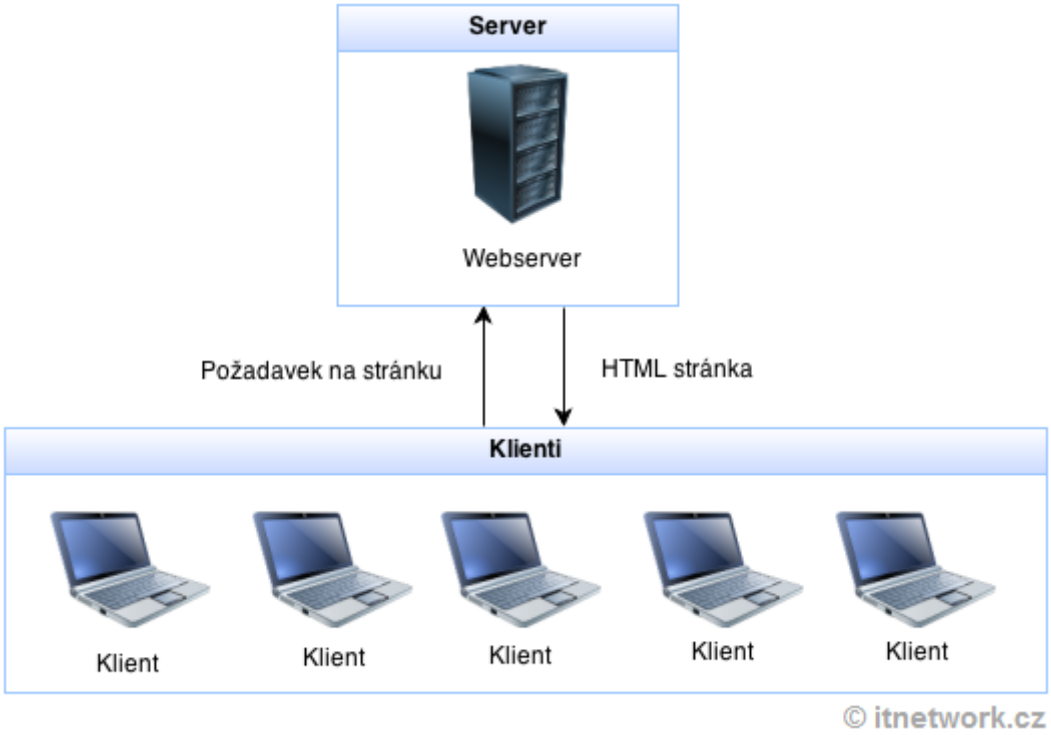
S rozšířením internetu nejprve vzniklo obrovské množství statických webových stránek.

Stránky sú textové soubory, kde pomocí značek označíte určitou část jako:

- nadpis,
- odstavec,
- Obrázek
- a podobně.

Výsledná stránka je (jak již bylo řečeno) statická. Nemůže se tedy měnit, je to pouze elektronický dokument, který můžeme jen číst.

HTML stránky jsou jednoduše uložené na serveru. Jakmile klient (uživatel s webovým prohlížečem) pošle požadavek na server, server mu jednoduše vrátí přesně tu stránku, co má uloženou. Této architektuře se říká klient-server.



Architektura nápadně podobá mainframu. Vracíme se tedy ke kořenům a získáváme následující výhody:

- Malá zátěž - Server pouze zasílá HTML stránky a již neřeší jejich zobrazení, ovládání klávesnice uživatele, jeho monitoru a podobně. To vše se děje na klientském počítači.

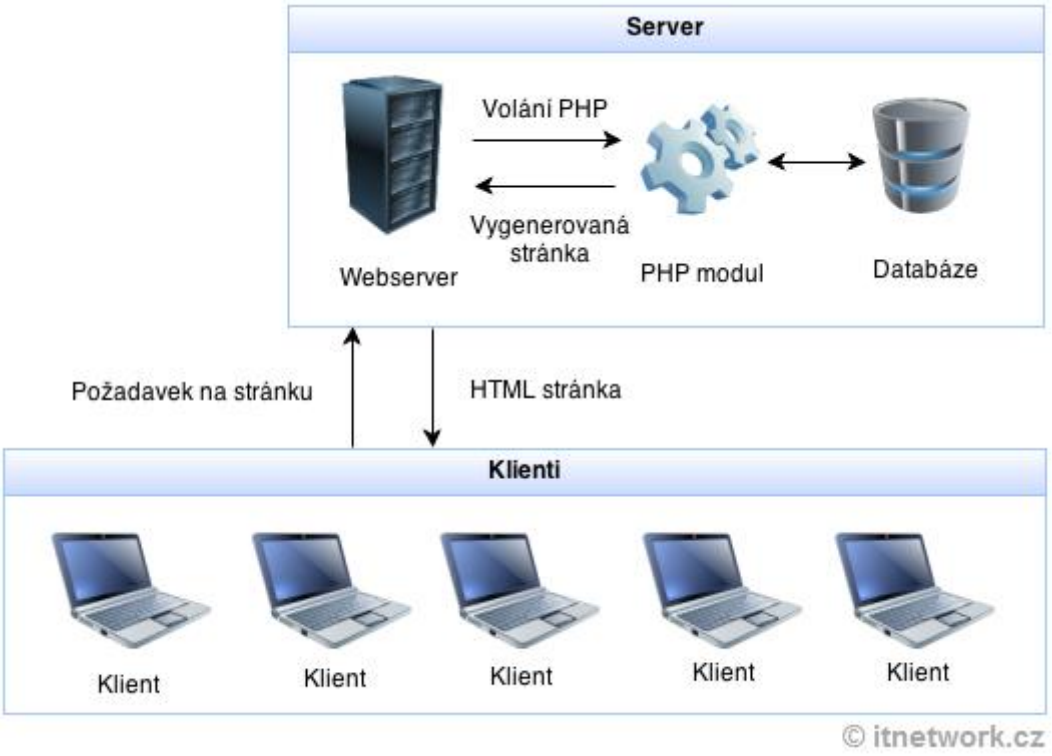
- Snadná správa - Jakmile změníme obsah na serveru (např. přidáme nějaký dokument), uvidí ho tam okamžitě všichni uživatelé.
- Vysoká bezpečnost - Pokud chceme zakázat přístup k nějakým dokumentům, klient se k nim prostě nemá jak dostat, jelikož jsou na serveru.

Nevýhodou je samozřejmě to, že server umí zasílat jen statické HTML stránky. Není způsob, jak serveru odeslat nějaká data a na jejich základě dostat výstup (třeba nechat na stránce vzkaz v diskuzi, hlasovat v anketě nebo zobrazit ve stránce kdo má dnes svátek).

Webové aplikace

V dnešní době jsme schopni dosáhnout toho, aby se webová stránka chovala úplně stejně, jako desktopová aplikace (třeba MS Word).Takúto aplikáciu nazývame **webová aplikace**.

Webové aplikace fungují tak, že se klient zeptá serveru na určitý dokument. Na serveru ale běží tzv. CGI skript, což je program, který dokáže vygenerovat do stránky to, co uživatel požaduje. Stránka tedy na serveru již neleží, ale je dynamicky vytvářena podle toho, co uživatel chce. Právě PHP je nejpoužívanějším CGI skriptovacím jazykem, ve kterém se webové aplikace píší.



Dynamický web pracuje nasledovne:

1. Uživatel vyťuká do prohlížeče URL adresu (třeba eshop.cz/tiskarna-epson-123) a tím pošle požadavek serveru
2. Server zavolá PHP modul

3. PHP modul se podívá, co uživatel chce (tady chce vypsat informace o tiskárně). Připojí se k databázi a načte data, která chce klient. Na základě dat vygeneruje webovou HTML stránku.
4. Hotová stránka je zaslána klientovi. Ten vidí již jen statickou webovou stránku, která však byla dynamicky vytvořená podle jeho požadavku.

Všechna data jsou v databázi a webová aplikace, zde v příkladu nějaký eshop s IT, má rozhraní, přes které může personál jednoduše přidávat nové produkty, upravovat jejich cenu a podobně. Je vám asi jasné, že kdyby data nebyla v databázi, ale každý produkt měl svou statickou HTML stránku, tak by bylo extrémně obtížné takovou spoustu stránek spravovat. Navíc by se potom nedalo pomocí skriptu třeba hledat podle ceny, psát k produktům komentáře a podobně.

Právě jsme si tedy popsali princip dynamického webu a také jeho příklad. Jaké jsou tedy výhody webových aplikací?

- Snadná správa - Novou verzi aplikace nahrajeme a v tom okamžiku ji používají úplně všichni.
- Vysoká bezpečnost - Web i databáze jsou na serveru a pokud neobsahuje nějaké bezpečnostní chyby, je velmi nesnadné aplikaci ukradnout.
- Vysoká uživatelská základna - Lidé jsou líní stahovat a instalovat. U webové aplikace jen kliknou na odkaz a již s ní pracují. Uživatelů, kteří by používali tu samou aplikaci, kdyby byla na desktopu místo na webu, by bylo podstatně méně. A právě kvůli uživatelům aplikace přeci píšeme. Ať jsou zadarmo nebo jsou placené, vždy chceme, aby je používalo co nejvíce lidí. Toto je hlavní důvod, proč se v poslední době dělá v podstatě veškerý software webový.
- Vysoká kompatibilita - Jelikož na web přistupujeme přes webový prohlížeč, vůbec nás nezajímá operační systém klienta, naše aplikace funguje prakticky všude, dokonce i na mobilu.
- Přetrvávají také výhody klientského počítače, tedy že server není zatěžován např. vlastním zobrazováním stránek, to dělá webový prohlížeč.

PHP /základy/

PHP je programovací jazyk, který pracuje na straně serveru. S PHP můžete ukládat a měnit data webových stránek. Původní význam zkratky PHP byl Personal Home Page. Vzniklo v roce 1996, od té doby prošlo velkými změnami a nyní tato zkratka znamená PHP: Hypertext Preprocessor.

Možnosti PHP

PHP . Umí ukládat, měnit a mazat data. Vše se odehrává na webovém serveru (kde jsou uloženy zdrojové kódy webových stránek). PHP skript se nejprve provede na serveru a potom odešle prohlížeči pouze výsledek (znamená to, že nejprve spočítá kolik je 300/30 a pak prohlížeči odešle jen číslo 10). Proto ve zdrojovém kódu najdete jen "10" (to je rozdíl oproti JavaScriptu, který počítá přímo v prohlížeči). Zdrojový kód PHP narozdíl od JavaScriptu a HTML v prohlížeči nezobrazíte.

Pomocí PHP je možné vytvořit :

- diskuzní fórum,
- knihu návštěv
- počítadlo,
- anketu,
- graf
- propojit vaše stránky s databázemi, např. MySQL

.

K čemu PHP?

Na webových stránkách se obvykle opakují některé části:

, hlavička s odkazy,

menu,

patička

. S PHP si můžete snadno vytvořit šablonu pro web, do které se budou vkládat soubory s menu, patičkou atd. Můžete tedy mít menu jen jednou zapsané a do dalších stránek ho pouze kopírovat. Až budete chtít menu změnit bude to nesmírně jednoduché. Více v článku PHP menu.

Soubory PHP

Webová stránka s prvky PHP má nejčastěji koncovku `.php`. Lze se však setkat i s dalšími koncovkami, např. `.phtml, php3, php4, php5`. Některé hostingy dle koncovky určovaly, pod jakou verzí PHP skript spustit (aktuální je 7). To je však velice výjímečné a v naprosté většině případů byste si měli vystačit s koncovkou `.php`.

Instalace

PHP je jazyk, který si nevystačí jen s prohlížečem určité verze (třeba jako HTML nebo JavaScript), ale je nutné ho na počítač nainstalovat.

- Localny server-localhost
- PHP server

. Základ tvoří webový server a knihovny

K podpoře PHP je třeba instalovat a konfigurovat server, obvykle Apache. Nejlepší je využít k

instalaci PHP program PHP Triad,

který vše sám nainstaluje.WinAMP

,,Xampp

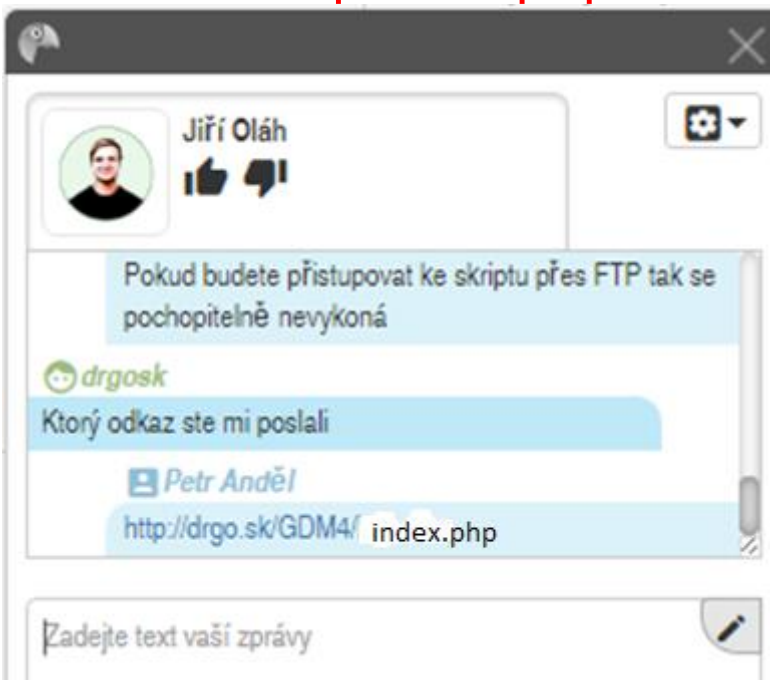
Webhosting s PHP

Ne každý webhosting zahrnuje podporu PHP. Potřebná podpora je u webhostingu nadstandardní službou za příplatek. Nicméně lze sehnat webhosting zdarma s podporou PHP (např. Webzdarma.cz, PHP 5). Při výběru webhostingu pro PHP stránky si pečlivě přečtěte, co nabídka zahrnuje.

Doufám, že vás PHP zaujala. V dalších článcích naleznete základy tohoto jazyka a jednoduché příklady.

Oficiální web PHP: php.net Další: Instalace PHP a PHP Triad
Také si můžete přečíst historii PHP
Srovnatelný programovací jazyk je: ASP

3.Spúšťanie PHP scriptov s php servera:



4.díl - Proměnné a typový systém v PHP

index.php

```
<!DOCTYPE html>
<html>
    <head>
       <meta charset = "UTF-8">
       <title>Uvítacia stránka</title>
    </head>
    <body>
        <?php
            echo("Víta Ťa server PHP5.3.0");
        ?>
    </body>
</html>
```

PHP skript vypadá jako běžná HTML stránka. Jelikož má příponu .php, může obsahovat části označené značkami (direktivou)

<?php ..... a ?>

Zde se může nacházet kód, který na serveru spustí PHP modul. Na toto místo potom vloží výsledek, který skript vygeneroval.

Příkaz echo do stránky vypíše zadaný text. Echo je **funkce** a každá funkce má za svým názvem vždy závorky, do kterých vložíme její vstupní parametry.

```
echo("vstupné parametre");
```

Pozn.: Zrovna u funkce echo lze závorky vynechat, ale u jiných to nelze. Přijde mi přehlednější psát je všude, než je někde mít a někde nemít.

Funkci echo() předáváme v parametru text. V programování se textu říká **textový řetězec**, anglicky **string**. Textové řetězce píšeme do uvozovek, ať už jednoduchých nebo dvojitých. Je to z toho důvodu, aby si PHP text nepletlo s ostatními příkazy. Když tedy napíšeme např.:

```
echo("echo");
```

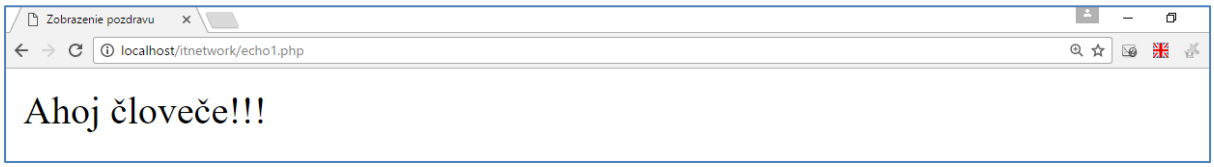
PHP ví, že je to druhé echo jen text a nebude si ho všímat (vypíše prostě text echo). Za každým příkazem vždy následuje středník. Příkazy píšeme na samostatné řádky. Syntaxe PHP vychází z céčka, ve kterém je PHP mimochodem naprogramované. Je samozřejmě podobná i dalším jazykům, co z céčka vycházejí, např. Javě nebo C# .NET. Pokud tyto jazyky znáte, budete mít výhodu. Pokud ne, vše si zde vysvětlíme od začátku.

Priklad na použitie príkazu echo:

echo1.php

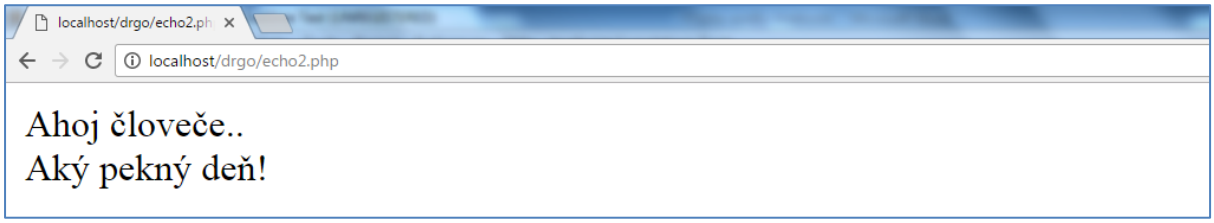
```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
</head>
<body>

<?php
        $pozdrav = "Ahoj človeče!!!";
```

```
        echo $pozdrav;
?>

</body>
</html>
```



echo2.php

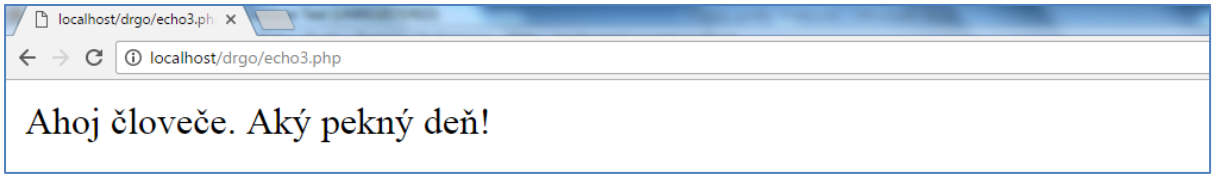
```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
</head>
<body>
<?php
        $text = "Ahoj človeče..";/*$ dolar je kombinácia pravý alt+ô */
        echo $text;
        echo "<br>Aký pekný deň!";
?>
</body>
</html>
```



echo3.php

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
</head>
<body>

<?php
        $text1 = "Ahoj človeče.";
        $text2 = "Aký pekný deň!";
        echo $text1 . " " . $text2;
?>
```

```
</body>
</html>
```

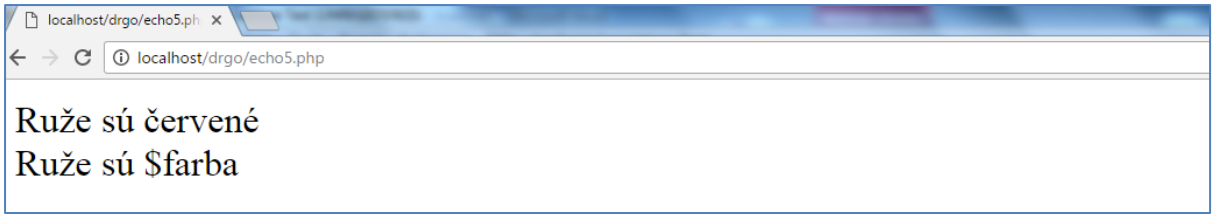


echo4.php

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
</head>
<body>
<?php
        echo 'Táto ','veta ','bola ','vytvorená ','s viacerých slov .';
?>
</body>
</html>
```

echo5.php

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
</head>
<body>
<?php
        $farba = "červené";
        echo "Ruže sú  $farba";
        echo "<br>";
        echo 'Ruže sú $farba';
?>
</body>
</html>
```



Proměnné

Slovo proměnná je miesto v paměti, kam si můžeme ukládat data a potom s nimi pracovat. Proměnné pojmenováváme libovolným názvem bez mezer a diakritiky. Před jejich název píšeme vždy dolar ($).*Poznámka (alt+ô)*

Uvnitř PHP části stránky si vytvoříme několik proměnných a naplníme je různými hodnotami:

```
$pozdrav = "Ahoj";
$vek = 15;
$pi = 3.14;
```

Všimněte si, že se proměnné vždy jmenují podle toho, co je v nich uloženo. Není nic horšího, než uložit např. věk do proměnné $a, pozdrav do $b a podobně. Ve svých skriptech byste se za chvíli nevyznali. Naopak pokud máte 2 čísla a pojmenujete je $a a $b, je to v pořádku. Obsah nějaké proměnné můžeme samozřejmě jednoduše vypsat pomocí funkce echo. Upravme náš kód:

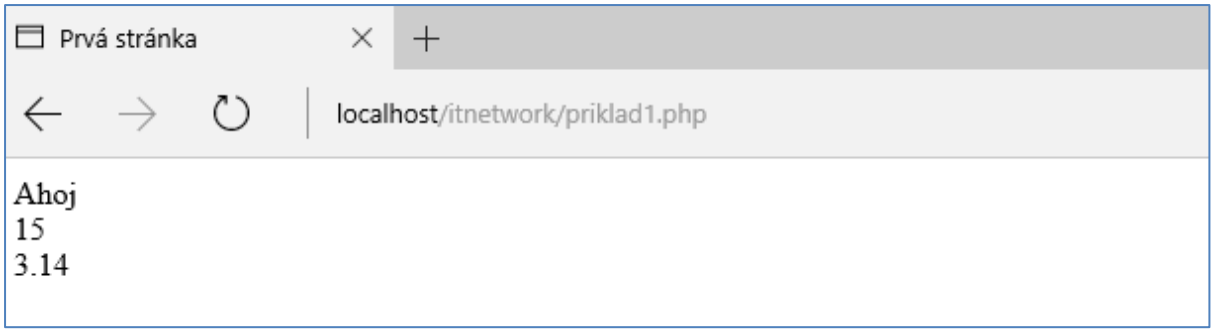
```
Kostra prikladu

pozdrav = "Ahoj";
$vek = 15;
$pi = 3.14;
echo($pozdrav);
echo('<br />');
echo($vek);
echo('<br />');
echo($pi);
echo('<br />');
```

premenne.php

```
<!DOCTYPE html>
<html lang="sk">
    <head>
       <meta charset="UTF-8">
        <title>Prvá stránka</title>
    </head>
    <body>
```

```
        <?php
          $pozdrav = "Ahoj";
          $vek = 15;
         $pi = 3.14;
         echo($pozdrav);
         echo('<br />');
         echo($vek);
         echo('<br />');
          echo($pi);
          echo('<br />');
        ?>
    </body>
</html>
```

Výstup



Určitě jste si všimli, že do uvozovek dáváme pouze text. Čísla a proměnné píšeme tak, jak jsou.

Datové typy

Každá proměnná je určitého typu, těmto typům se říká datové. Ve skriptu jsme si vytvořili proměnné třech základních datových typů. Pozdrav je typu **string**, $vek je typu **int**, což je celé číslo. $pi je potom typu **double** (někdy můžete narazit i na typ float, v PHP označují oba typy to samé, tedy desetinné číslo).

PHP je tzv. dynamicky typovaný jazyk. To znamená:

- že datové typy nemusíme u proměnných zadávat (jako třeba v jazyce C), ale PHP si typ podle obsahu proměnné nastaví samo.
- Mezi typy také PHP samo převádí. Teoreticky nemusíme ani vědět o tom, že proměnná nějaký datový typ má, prakticky bychom se však někdy mohli docela divit, když by PHP samo převedlo něco tak, jak jsme to nečekali.

Sčítání čísel a spojování řetězců

V PHP můžeme mezi čísly samozřejmě používat základní početní operace, kulaté závorky a ostatní proměnné:

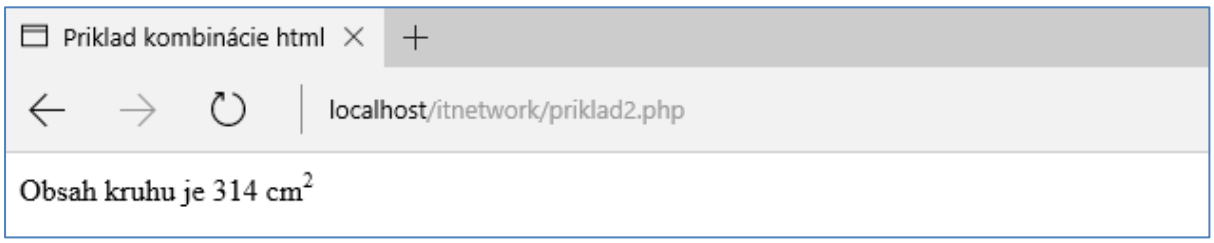
Kostra príkladu

```
$r = 10;
$obsah = 3.14 * $r * $r;
echo("Obsah kruhu je $obsah cm<sup>2</sup>");
```

obsahKruhu.php

```
<!DOCTYPE html>
<html lang="sk">
    <head>
       <meta charset="UTF-8">
        <title>Príklad kombinácie html a php scriptu</title>
    </head>
    <body>
        <?php
                $r = 10;//Polomer kruhu
                $obsah = 3.14 * $r * $r;
                echo("Obsah kruhu je $obsah cm<sup>2</sup>");
        ?>
    </body>
</html>
```

Výstup



Kód výše vytvoří :

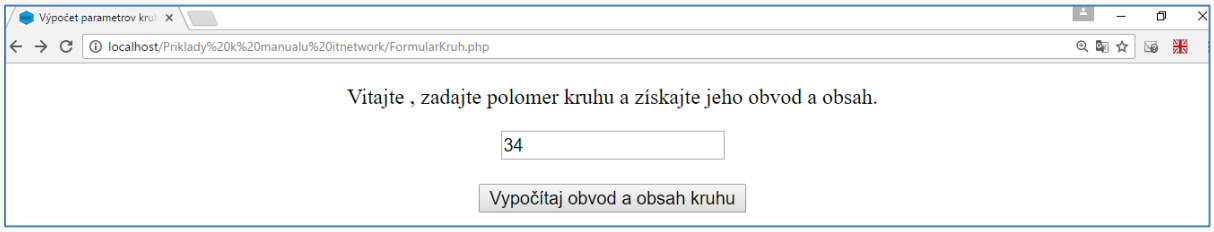
- proměnnou **$r** s hodnotou 10.
- Tato hodnota se využije při zadávání obsahu proměnné **$obsah**.
- Proměnná **$obsah** se vypíše spolu s dalším textem funkcí echo.

Vidíte, že do řetězce můžeme jednoduše vložit i obsah proměnné. Funguje to však pouze s dvojitými uvozovkami, s jednoduchými by se vypsal řetězec tak, jak je napsaný a proměnná by se do něj nevložila. Horního indexu u čtvečních centimetrů jsme dosáhli

pomocí HTML tagu sup. Poloměr by se v reálné aplikaci samozřejmě zadal nějakým formulářem, aby měla aplikace nějaký význam. Teraz nasleduje prečastné použitie asociatívneho poľa.Tak to teraz skusíme:

FormularKruh.php

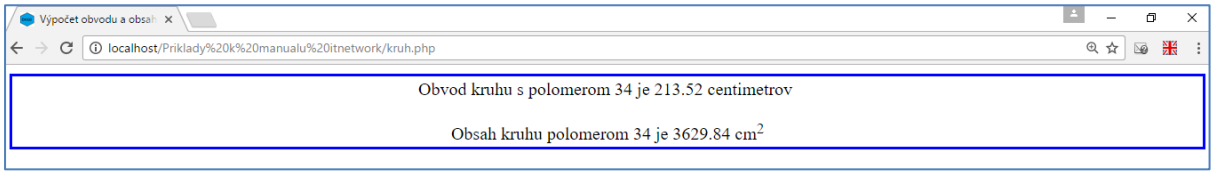
```
<!DOCTYPE html>
<html lang="sk">
    <head>
        <meta charset="UTF-8">
        <title>Výpočet parametrov kruhu</title>
        <link href='logo.png' rel='shortcut icon' type='image/png'>
    </head>
    <body>
        <div align=center>
        <p>Vitajte , zadajte polomer kruhu a získajte jeho obvod a
obsah.</p>

        <form method="POST" action="kruh.php">
            <input name="polomer" type="number" /><br /><br />
            <input type="submit" value="Vypočítaj obvod a obsah kruhu" />
        </form>
        </div>
    </body>
</html>
```



kruh.php

```
<!DOCTYPE html>
<html>
    <head>
    <meta charset="UTF-8">
       <title>Výpočet obvodu a obsahu kruhu</title>
       <link href='logo.png' rel='shortcut icon' type='image/png'>
    <style>
        table {
            width: 100%;
            border: 3px solid blue;
            text-align: center;
                }
    </style>
```

```
    </head>
    <body>
        <table><tr><td>
         <?php
            $r = $_POST['polomer'];
            $obvod = 2*3.14*$r ;
            $obsah = 3.14 * $r * $r;
            echo("Obvod kruhu s polomerom $r  je $obvod centimetrov
<br><br>");
           echo("Obsah kruhu polomerom $r je $obsah cm<sup>2</sup>");
        ?>
        </td></tr></table>
    </body>
</html>
```



Již jsme si říkali, že PHP samo převádí mezi různými datovými typy. Vyzkoušejme si to:

```
$a = 10;
$b = "20";
$c = $a + $b;
echo($c);
```

Výstupem bude:

```
30
```

Ačkoli je proměnná $b zadána jako text, PHP ji převede na číslo jakmile zjistí, že jí chceme k něčemu přičíst. Stejného výsledku bychom dokonce dosáhli i s tímto zadáním:

```
$a = 10;
$b = "20 let mi je";
$c = $a + $b;
echo($c);
```

To už je trochu divoké a rozhodně to nebudeme dělat.

Stejně, jako můžeme sčítat čísla, můžeme **spojovat** textové řetězce. Cizím slovem se tomu říká konkatenacia (anglicky concatenation nebo zkráceně concat). Slouží k tomu operátor tečka (**.**):

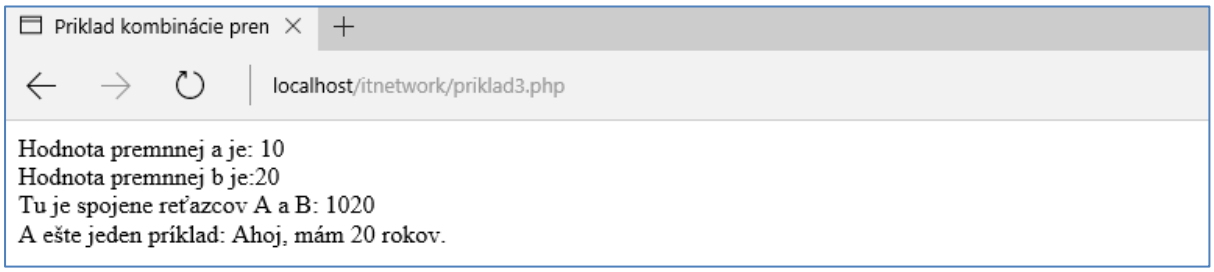
Kostra príkladu

```php
$a = 10;
$b = 20;
$veta = "Ahoj, mám";

$sucet = $a + $b;
$spojenie = $a . $b;
echo("Tu je súčet čísel A a B: $sucet");
echo('<br />');
echo("Tu je spojenie reťazcov A a B: $spojenie");
echo('<br />');
echo('A ešte jeden príklad: ');
echo($veta . " " . $b . " rokov.");
```

kombinaciaPremennych.php

```php
<!DOCTYPE html>
<html lang="sk">
    <head>
       <meta charset="UTF-8">
        <title>Priklad kombinácie premenných</title>
    </head>
    <body>
        <?php
         $a = 10;
         $b = 20;
          echo("Hodnota premennej a je: $a");
          echo('<br />');
          echo("Hodnota premnnej b je:$b");
         $veta = "Ahoj, mám ";
         $sucet = $a + $b;
         $spojenie = $a . $b;
         echo('<br />');
         echo("Tu je spojene reťazcov A a B: $spojenie");
         echo('<br />');
         echo('A ešte jeden príklad: ');
         echo($veta . " " . $b . " rokov.");
        ?>
    </body>
</html>
```

Výstup:



Z výstupu programu vidíme rozdíl mezi sčítáním čísel a spojováním řetězců. Všimněte si, že jsme stále v HTML, pokud chceme nový řádek, vložíme do stránky prostě HTML tag pro odřádkování.

5.díl - Textové řetězce podruhé a pole v PHP

Zápisy řetězců a escapování

V minulém dílu jsme si ukazovali, jak zapisovat textové řetězce. Víme, že je můžeme psát do apostrofů nebo uvozovek. Abychom mohli stavět na nějakých základech, vysvětleme si rozdíly v těchto dvou zápisech.

Apostrofy

Pokud napíšeme textový řetězec do apostrofů (jednoduchých uvozovek), vloží se do něj přesně ten text, který je uveden:

Kostra príkladu:

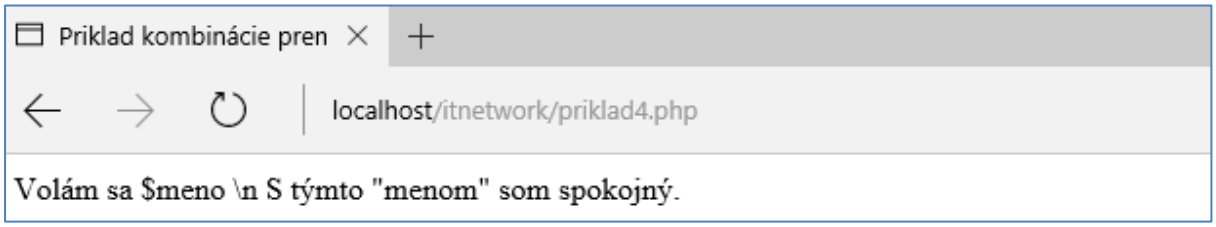
```
$meno = 'Karol';
$text = 'Volám sa  $meno \n S týmto "menom" som  spokojný.';
echo($text);
```

Priklad4.php

```
<!DOCTYPE html>
<html lang="sk">
    <head>
       <meta charset="UTF-8">
        <title>Priklad kombinácie reťazcov</title>
    </head>
    <body>
        <?php
        $meno = 'Karol';
```

```
        $text = 'Volám sa  $meno \n S týmto "menom" som  spokojný.';
        echo($text);

        ?>
    </body>
</html>
```

Výstup:



Vidíme, že si PHP vůbec nevšímá znaku dolaru, dvojitých uvozovek ani zpětného lomítka (to si vysvětlíme dále).

Pokud chceme do řetězce v apostrofech vložit obsah nějaké proměnné, nemáme jinou možnost, než použít operátor **bodky**:

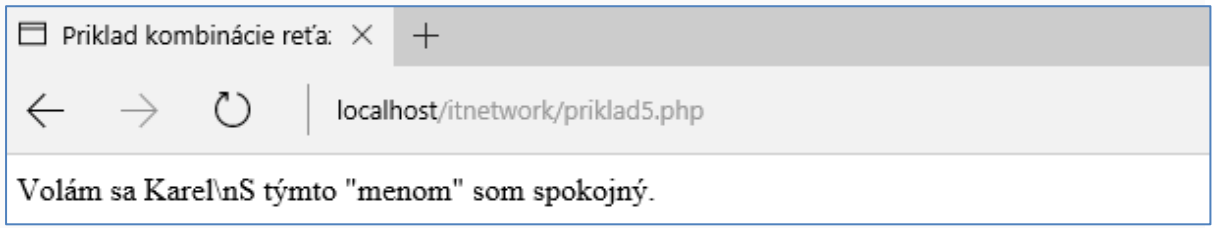
Kostra príkladu

```
$meno = 'Karol';
$text = 'Volám sa  ' . $meno . ' \n S týmto "menom" som spokojný.';
echo($text);
```

Príklad5.php

```
!DOCTYPE html>
<html lang="sk">
    <head>
       <meta charset="UTF-8">
        <title>Priklad kombinácie reťazcov</title>
    </head>
    <body>
        <?php
        $meno = 'Karol';
        $text = 'Volám sa  ' . $meno .'\nS týmto "menom" som spokojný.';
        echo($text);
        ?>
    </body>
</html>
```

Výstup



Uvozovky

Uvozovky jsou v PHP chytřejší apostrofy a do řetězce zapsaném pomocí uvozovek můžeme jednoduše vkládat proměnné:

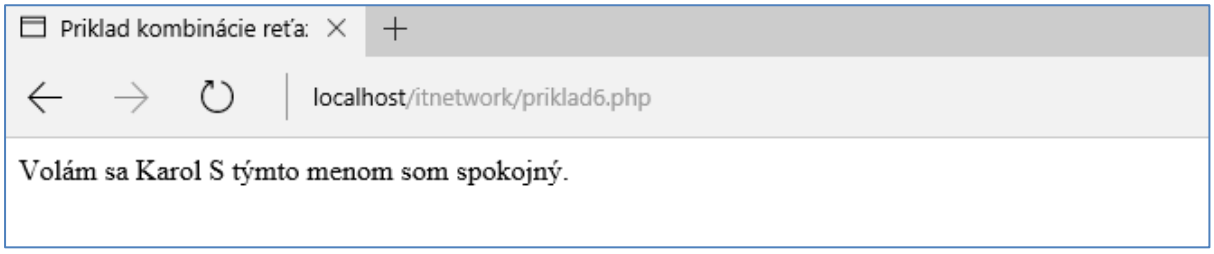
Kostra príkladu

```
$meno = 'Karol';
$text = "Volám sa  $meno \n S týmto menom som  spokojný.";
echo($text);
```

Príklad6.php

```
<!DOCTYPE html>
<html lang="sk">
    <head>
       <meta charset="UTF-8">
        <title>Priklad kombinácie reťazcov</title>
    </head>
    <body>
        <?php
        $meno = 'Karol';
        $text = "Volám sa  $meno \n S týmto menom som  spokojný.";
        echo($text);
        ?>
    </body>
</html>
```

Výstup bude následující:



Do textu se vložila hodnota proměnné $meno. Také vidíte, že se nevypsala sekvence \n. To z toho důvodu, že pomocí lomítka se v uvozovkách zapisují speciální znaky. Z těch

základních je \n odřádkování a \t tabulátor. V HTML se odřádkování neprojeví, ale když se podíváte do zdrojového kódu vygenerované stránky, tak ho tam uvidíte.

Pole

Představte si, že ve svém programu potřebujete uložit více proměnných stejného typu. Např. česká jména měsíců, 10 známek ze školy, platy 100 uživatelů nebo databázi tisíců firem. Asi tušíte, že bude nějaká lepší cesta, než začít zběsile bušit proměnné znamka1, znamka2, znamka3... A co kdyby se program rozšířil a známek by bylo tisíc? A jak by se potom počítal průměr z těchto známek?

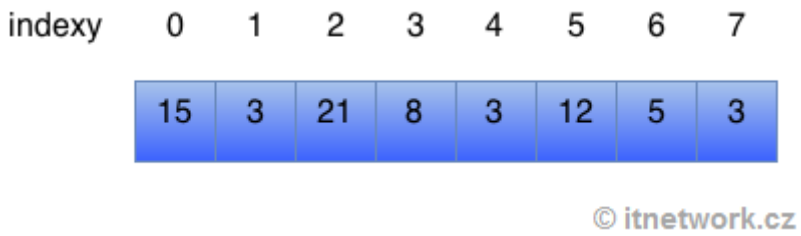
Pro uložení více proměnných stejného typu nám PHP nabízí datovou strukturu, které se říká pole. V PHP jsou 2 typy polí. Dnes si popíšeme první z nich.

1.Číselně indexovaná pole

Číselně indexované pole si můžeme představit jako řadu priehradiek někde v paměti počítače (ono mimochodem vnitřně opravdu tak vypadá).

Přihrádky jsou číslované vždy od nuly a jejich číslům se říká indexy. V programování víceméně vše začíná od nuly, ne nadarmo se říká, že první dítě programátora je nulté 🙂 Přihrádek můžeme mít kolik chceme a můžeme si do nich uložit proměnné libovolného typu a to dokonce i tak, že bude v jedné přihrádce třeba číslo a v druhé text. V praxi by to ovšem nemělo valný smysl.

Na obrázku níže vidíte znázorněné pole, ve kterém je uloženo 8 čísel:



Zkusme si něco podobného naprogramovat, udělejme si zmíněné pole známek. Známky opět napíšeme pevně do skriptu, jelikož ještě neumíme formuláře. To napravíme hned v příští lekci 🙂

Pole si uložíme do klasické proměnné. Pojmenujeme ji tak, co v ní je a v množném čísle, tedy v našem případě $znamky. Pro samotnou definici pole používáme klíčové slovo array().

```php
$znamky = array();
```

Nyní je v proměnné **$znamky** prázdné pole. Pokud používáte PHP 5.4 nebo novější, můžete pole zapsat i takto:

```php
$znamky = [];
```

Tento zápis je kratší a obecně lepší, ale já se budu držet starší definice, jelikož někteří čtenáři mohou používat starší webhostingy.

Vkládání prvků do pole

Nyní do pole přidejme několik známek. Jak je zvykem, můžeme to udělat několika způsoby.

Nový prvek na konec pole přidáme takto:

```php
$znamky[] = 1;
$znamky[] = 2;
$znamky[] = 5;
```

Výsledné pole tedy vypadá takto:

```
1, 2, 5
```

Samotné echo() vám pole moc dobře nevypíše, ale můžete si ho nechat vypsat pomocí funkce print_r():

```php
print_r($znamky);
```

Stejně tak můžeme vložit prvek na konkrétní index v poli. Pokud prvek na tomto indexu už je, prostě se přepíše.

```php
$znamky[0] = 1;
$znamky[1] = 3;
$znamky[2] = 5;
$znamky[1] = 2; // Přepíše prvek na indexu 1 (3) na dvojku
```

Poslední řádek vloží na index 1 prvek 2, tím přepíše hodnotu 3, která se na tento index již vložila na druhém řádku. Nemá to samozřejmě žádný smysl, jen ten, že si to vyzkoušíme 🙂 To za dvěma lomítky je tzv. komentář. PHP si těchto textů nevšímá a slouží pro lepší orientaci programátorů v kódu. Určitě je dobrý nápad komentáře používat. Dají se zapsat i na více řádků a to takto:

```php
/*
Tento text PHP ignoruje a platí
až do ukončení komentáře. Můžeme si tak popsat
složitější části programu, abychom nezapomněli
jak jsme to mysleli, až ke kódu přijdeme za pár měsíců :)
*/
```

Pokud známky předem víme, můžeme pole vytvořit rovnou se zadáním jeho obsahu:

```php
$znamky = array(1, 2, 3, 4, 2, 2, 1, 3, 2, 5);
```

Čtení hodnot z pole

Pokud z pole potřebujeme načíst nějakou hodnotu, uděláme to pomocí indexu:

```php
echo('Třetí známka v poli je ' . $znamky[2]);
```

Výstup:

```
Třetí známka v poli je 3
```

Praktické ukázky použití polí

PHP nám nabízí pro práci s poli spoustu funkcí. I když si je popíšeme až v dalších dílech, udělejme si jen takové dvě malé ochutnávky.

Průměr známek

Pomocí funkcí array_sum() a count() spočítáme průměr zadaných známek:

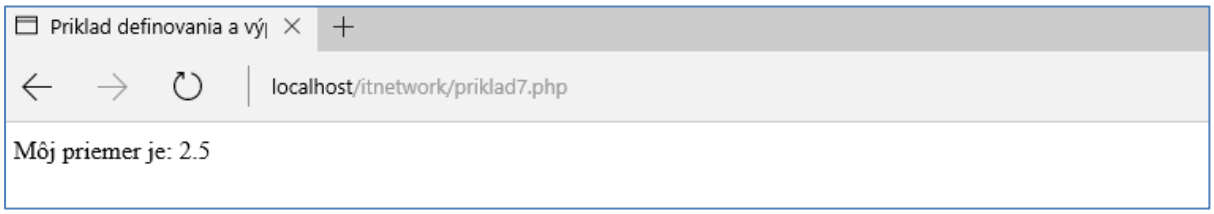
Kostra príkladu

```php
$znamky = array(1, 2, 3, 4, 2, 2, 1, 3, 2, 5);
$priemer = array_sum($znamky) / count($znamky);
echo('Môj priemer je: ' . $priemer);
```

Priklad7.php

```php
<!DOCTYPE html>
<html lang="sk">
    <head>
       <meta charset="UTF-8">
        <title>Priklad definovania a výpisu poľa</title>
    </head>
    <body>
        <?php
         $znamky = array(1, 2, 3, 4, 2, 2, 1, 3, 2, 5);
                $priemer = array_sum($znamky) / count($znamky);
                echo('Môj priemer je: ' . $priemer);
        ?>
    </body>
</html>
```

Výstup programu:



Asi si dokážete představit, jak špatně by byl program čitelný, kdybychom pole neznali a psali $sucet = $znamka1 + $znamka2 + $znamka3... Nyní jsme jen jednoduše vydělili součet prvků v poli jejich počtem.

Slovenký dátum

Jak již bylo řečeno, do pole si můžeme uložit prakticky cokoli, třeba názvy měsíců. Můžeme potom vypsat uživateli aktuální datum a to dokonce slovensky:

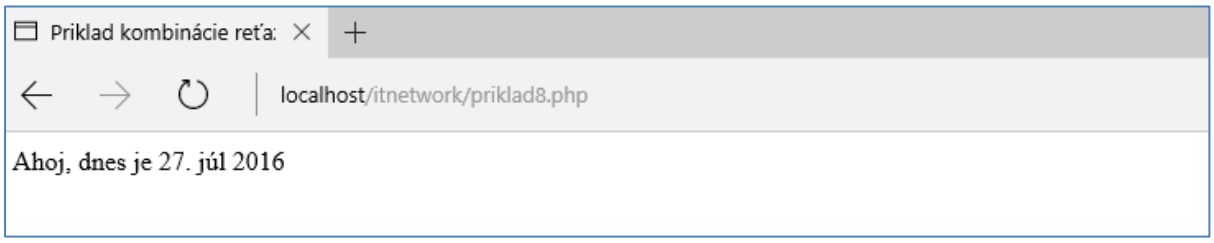
Kostra príkladu

```
$mesiace = array('január', 'február', 'marec', 'apríl', 'máj', 'jún',
    'júl', 'august', 'september', 'október', 'november', 'december');
$den = date('d');
$mesiac = date('m');
$mesiacSlovami = $mesiace[$mesiac - 1];
$rok = date('Y');
echo("Ahoj, dnes je $den. $mesiacSlovami $rok");
```

Priklad8.php

```
<!DOCTYPE html>
<html lang="sk">
    <head>
       <meta charset="UTF-8">
        <title>Priklad kombinácie reťazcov</title>
    </head>
    <body>
        <?php
       $mesiace = array('január', 'február', 'marec', 'apríl', 'máj',
'jún',
    'júl', 'august', 'september', 'október', 'november', 'december');
     $den = date('d');
     $mesiac = date('m');
     $mesiacSlovami = $mesiace[$mesiac - 1];
     $rok = date('Y');
     echo("Ahoj, dnes je $den. $mesiacSlovami $rok");
        ?>
    </body>
</html>
```

Výstup



Vysvetlenie scriptu:

- V programu vytvoříme pole názvů měsíců.
- Dále si pomocí funkce **date()** zjistíme číslo dne, měsíce a roku.
- Do proměnné **$mesiacSlovami** si budeme chtít uložit aktuální měsíc z pole. Když je október (měsíc 10), máme v proměnné $mesiac hodnotu 10.
- Prvek v poli s hodnotou "október" má však index 9, jelikož jsou číslované od nuly. Proto sáhneme na přihrádku s indexem $mesiac - 1.
- Nakonec jen jednoduše vypíšeme obsah proměnných. Kompletní práci s datem a časem se také naučíme později.

Cvičení k 4. lekci PHP

Následující 3 cvičení vám pomohou procvičit znalosti programování v PHP z minulé lekce.

Jednoduchý příklad

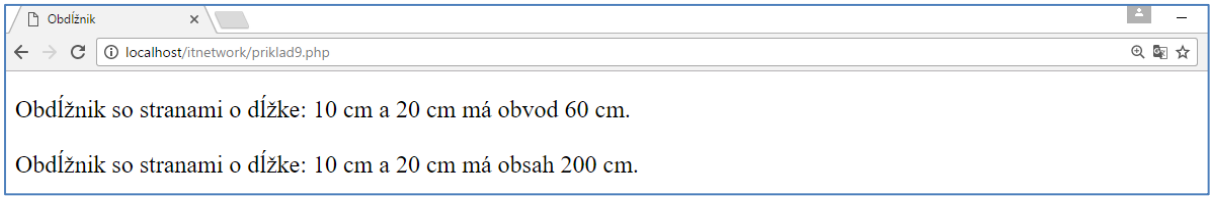
Vytvořte skript, který si založí proměnné **$a** a **$b**. Do těchto proměnných vloží délky stran obdélníka (hodnoty si vymyslete) a potom vypíše jeho obvod a obsah. Pomocí spojování řetězců dosáhněte následujícího výstupu:

Priklad9.php

```
<!DOCTYPE html>

<html lang="sk">
<head>
        <meta charset="utf-8" />
        <title>Obdĺžnik</title>
</head>

<body>
        <p>
        <?php
                        $a = 10;
```

```
                        $b = 20;
                        $obsah = $a * $b;
                        $obvod = 2 * ($a + $b);
                        echo("Obdĺžnik so stranami o dĺžke: $a cm a $b cm má
obvod $obvod cm.<br><br>");
            echo("Obdĺžnik so stranami o dĺžke: $a cm a $b cm má obsah
$obsah cm.");

        ?>
        </p>
</body>
</html>
```

Výstup



Ten istý priklad s použitím asociatívneho poľa

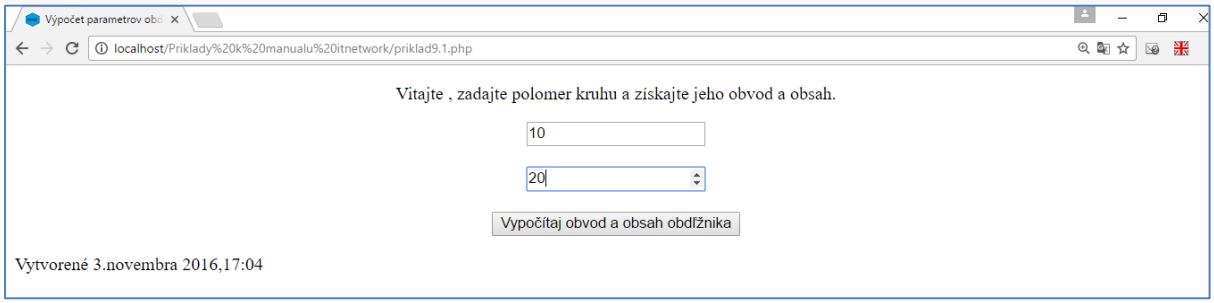
Priklad9.1.php

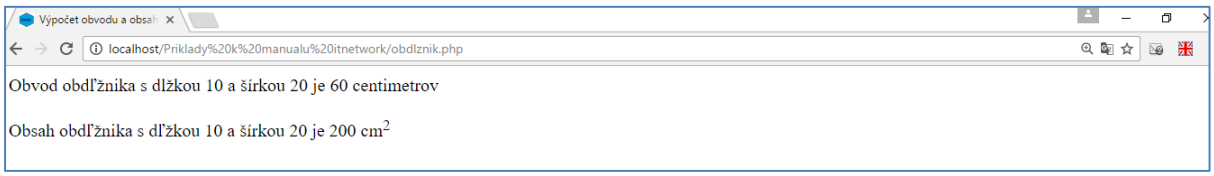
```
<!DOCTYPE html>
<html lang="sk">
    <head>
        <meta charset="UTF-8">
        <title>Výpočet parametrov obdĺžnika</title>
        <link href='logo.png' rel='shortcut icon' type='image/png'>
    </head>
    <body>
        <div align=center>
        <p>Vitajte , zadajte polomer kruhu a získajte jeho obvod a
obsah.</p>

        <form method="POST" action="obdlznik.php">
            <input name="dlzka" type="number" /><br /><br />
            <input name="sirka" type="number" /><br /><br />
            <input type="submit" value="Vypočítaj obvod a obsah obdĺžnika"
/>
        </form>
        </div>
        <p>Vytvorené 3.novembra 2016,17:04<p>
    </body>
```

```
</html>
```

obdlznik.php

```
<!DOCTYPE html>
<html>
    <head>
    <meta charset="UTF-8">
       <title>Výpočet obvodu a obsahu obdlžnika zo zadaných hodnôt</title>
       <link href='logo.png' rel='shortcut icon' type='image/png'>

    </head>
    <body>
        <?php

            $a = $_POST['dlzka'];
            $b = $_POST['sirka'];
            $obvod = 2*($a+$b) ;
            $obsah = $a * $b;
            echo("Obvod obdĺžnika s dlžkou $a a šírkou $b je $obvod
centimetrov <br><br>");
          echo("Obsah obdĺžnika s dĺžkou $a  a šírkou $b je $obsah
cm<sup>2</sup>");
        ?>
    </body>
</html>
```





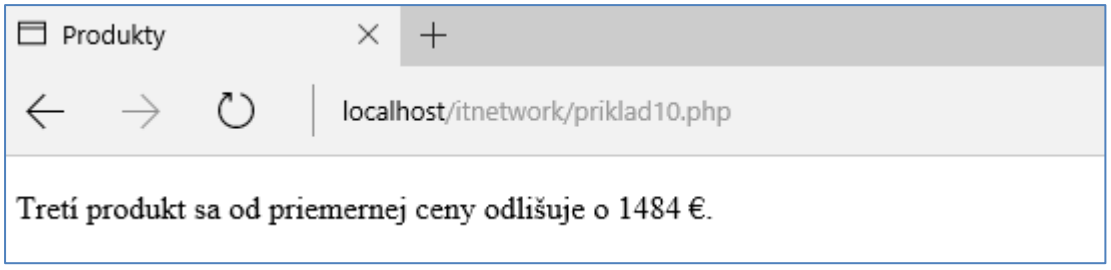
Středně pokročilý příklad

Napište skript, který eviduje produkty s následujícími cenami: 3500€, 2800€, 1280€, 1920€, 4320€. Spočítejte odchylku 3. produktu od průměrné ceny. K řešení úlohy využijte pole.

Príklad10.php

```
!DOCTYPE html>

<html lang="sk">
<head>
        <meta charset="utf-8" />
        <title>Produkty</title>
</head>

<body>
        <p>

                <?php



                        $ceny = array(3500, 2800, 1280, 1920, 4320);
                        $priemer = array_sum($ceny) / count($ceny);
                        $odchylka = $priemer - $ceny[2];
                        echo("Tretí produkt sa od priemernej ceny odlišuje o
$odchylka €.");
                ?>

        </p>
</body>
</html>
```

výstup



Pokročilý příklad

V čínské astrologii je každému roku přiřazeno jedno ze zvířecích znamení. Ta jsou: podkana, byvola, tygra, zajaca, draka, hada, koňa, kozy, ovce, opice, kohúta, psa a prasaťa. Vytvořte skript, který podle roku, uloženého v proměnné $rok, vypíše znamení pro tento rok. Rok 2014 je rok koně, 2015 je tedy kozy a tak dále. Znamení se cyklí

stále dokola, my ovšem ještě neumíme podmínky, tak se musíme spokojit s tím, že naše aplikace bude fungovat jen na 12 let dopředu.

Priklad11.php

```
<!DOCTYPE html>
<html lang="sk">
<head>
        <meta charset="utf-8" />
        <title>Zverokruh</title>
</head>
<body>
        <p>
                <?php


                        $znamenie = array('koňa', 'kozy', 'ovce', 'opice',
'kohúta', 'psa', 'prasaťa',
                                                        'podkana', 'byvola',
'tygra', 'zajaca', 'draka', 'hada');
                        $rok = date('Y'); // Tuto hodnotu môžeme upravit
nebo vložit podle aktuálního dátumu funkciou date('Y');
                        $aktualneZnamenie = $znamenie[$rok - 2014];
                        echo("Je rok $aktualneZnamenie");
                ?>

        </p>
</body>
</html>
```

Ukázka obrazovky programu:

5. díl - Asociativní pole v PHP a obsluha formulářů

2.Asociativní pole

Druhý typ polí v PHP je tzv. pole asociativní. Funguje úplně stejně, jako číselně indexované, ale indexy již nejsou čísla, ale textové řetězce. **Indexům se v tomto typu polí říká klíče**.

Asociativní pole definujeme podobně jako číselně indexované, jen kromě hodnot zadáváme i klíče. K tomu používáme operátor dvojité šipky (=>):

```
$oblubeneVeci = array(
        'homer' => 'siska',
        'marge' => 'trubka',
        'bart' => 'prak',
        'liza' => 'kniha',
        'meggie' => 'dudlík',
);
```

Zápis pole jsme rozdělili kvůli přehlednosti do více řádků, ale šlo by to i v jednom. V poli máme 5 hodnot: 'siska', 'trubka', 'prak', 'kniha', 'dudlík'. Každá hodnota patří nějakému klíči ('homer', 'marge', 'bart', 'liza', 'meggie'). Hodnoty přiřadíme ke klíči pomocí šipky a oddělujeme čárkou, která se většinou píše i za poslední položkou. Nezapomenete ji tak napsat až do pole budete přidávat další prvek.

Mimochodem, všimněte si, že pokud vytváříme proměnnou, jejíž název zahrnuje více slov, použijeme tzv. velbloudí notaci. První písmeno malé a každé počáteční písmeno dalšího slova velké. Lidé v PHP píší různě, ale tato konvence je bezesporu nejlepší.

Pro práci s asociativním polem platí to samé, jako jsme si ukazovali minule u pole číselně indexovaného:

```
echo('Homer má rád: ' . $oblubeneVeci['homer']);
```

Místo abychom napsali $oblibeneVeci[0], použijeme textový klíč. Obrovskou výhodou je přehlednost. Vidíme, co z pole vytahujeme, na rozdíl od číselného indexu, který nám mnohdy nic neříká.

Každé číselně indexované pole lze zapsat jako pole asociativní a to takto:

Kostra príkladu

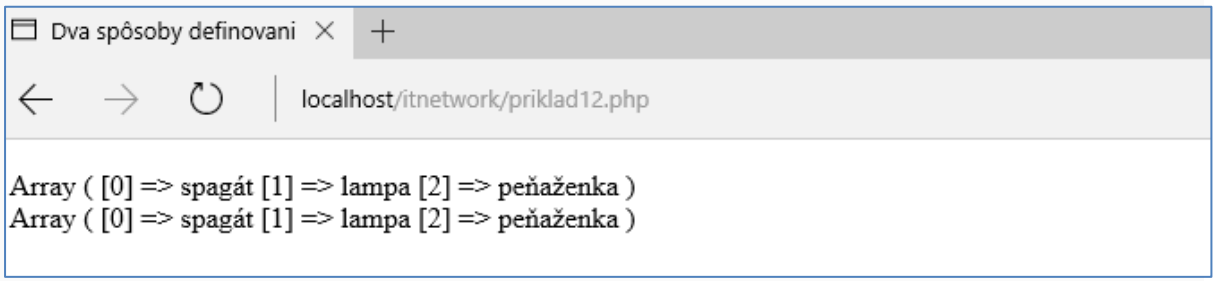
```
$batoh = array('spagat', 'lampa', 'penazenka');

$batoh2 = array(
        0 => 'spagat',
        1 => 'lampa',
        2 => 'penazenka',
);
```

```php
print_r($batoh);
print_r($batoh2);
```

Priklad12.php

```php
<!DOCTYPE html>
<html lang="sk">
<head>
        <meta charset="utf-8" />
        <title>Dva spôsoby definovania asociatívneho poľa</title>
</head>
<body>
        <p>
                <?php
                //Prvý spôsob
                $batoh = array('spagát', 'lampa', 'peňaženka');
        //Druhý spôsob
                 $batoh2 = array(
                 0 => 'spagát',
                 1 => 'lampa',
                 2 => 'peňaženka',
                                );

              print_r($batoh);//Preto print_r lebo echo nezobrazí pole
              echo "<br />";
              print_r($batoh2);

                ?>

        </p>
</body>
</html>
```

Výstup



Z výpisu vidíme, že pole $batoh a $batoh2 jsou naprosto stejná. PHP totiž vnitřně zná jen pole asociativní a když založíme číselně indexované, jednoduše vytvoří indexy podle pořadí položek.

Pozn.: Ve většině jazyků je pole omezené svou velikostí a když se jednou vytvoří, nelze do něj prvky přidávat. V PHP tomu tak není, s polem si můžete dělat úplně co chcete.

Zpracování formulářů v PHP

Konečně se dostáváme k něčemu zajímavému. Pravé aplikace přeci reagují na vstup od uživatele, zkusíme si takovou aplikaci vytvořit - naprogramujeme si jednoduchou webovou kalkulačku.

Webové aplikaci lze předat vstup pomocí dvou metod - GET a POST. HTTP protokol zná ještě několik dalších metod (REST), ale prohlížeče je nepodporují. Data dorazí do PHP skriptu vždy v asociativním poli, které se jmenuje podle metody, kterou přišla.

Vytvořte si nějakou složku pro novou aplikaci (třeba kalkulacka).

1.Metoda GET

Vytvořte si soubor kalkulacka.php. Pokud budeme tomuto skriptu chtít předat nějaká data metodou GET, zadáme je do URL adresy pomocí tzv. query stringu. Query string začíná otazníkem (?) a jednotlivé parametry jsou oddělené ampersandy (&). Mezi názvem parametru a jeho hodnotou je rovná se (=).

Do URL adresy bychom tedy zadali toto:

```
localhost/itnetwork/kalkulacka.php?cislo1=10&cislo2=20
```

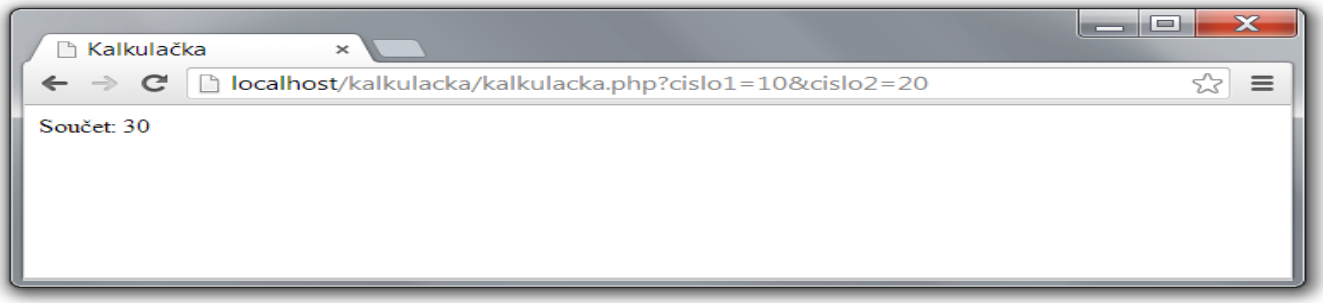
Skriptu předáváme metodou GET dvě proměnné. První se jmenuje cislo1 a má hodnotu 10, cislo2 má hodnotu 20.

Obsah skriptu by mohl být následující:

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
        <title>Kalkulačka</title>
    </head>
    <body>
        <?php
            $soucet = $_GET['cislo1'] + $_GET['cislo2'];
            echo("Součet: $soucet");
        ?>
    </body>
</html>
```

Skript pracuje s tzv. **superglobálním polem $_GET**. Superglobálním proto, že je přístupné odkudkoli. Všechny proměnné, které byly skriptu odeslány v query stringu, nalezneme v tomto poli. Pole je asociativní, jako klíč zadáme název proměnné a dostaneme její hodnotu. Součet hodnot jednoduše vypíšeme.

Výsledek:



Metoda GET slouží spíše pro získávání podstránek webu a pro kalkulačku se příliš nehodí.

Otestovanie znalostí z PHP

Správna odpoveď	
1B	**11C**
2C	**12A**
3B	**13A**
4B	**14A**
5A	**15B**
6B	**16A**
7A	**17B**
8B	
9B	
10A	

HODNOTENIE

Body	Známka
17-16	**1**
15-14	**2**
13-9	**3**
8-5	**4**
<5	**5**

2.Metoda POST

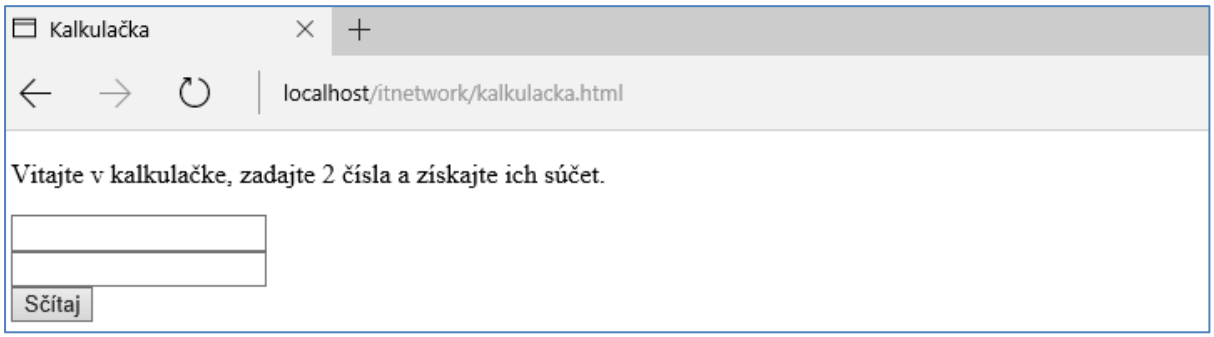
Metoda POST se používá při odesílání formulářů. Naše aplikace se bude nyní skládat ze dvou souborů. V prvním bude formulář, kam uživatel zadá 2 čísla do textových polí a odešle tlačítkem "sčítaj". Data se odešlou druhému souboru, což bude PHP skript, který provede výpočet.

kalkulacka.html-keď tento súbor otvoríme spustí sa script sucet.php

Soubor obsahuje formulář. Můžeme mu dát koncovku PHP, ale není to nutné, žádné PHP bloky zde nejsou. Pokud soubor pojmenujete index, zobrazí se samozřejmě jako výchozí stránka, když do složky kalkulacka přistoupíte. Soubor bude vypadat asi takto:

```
<!DOCTYPE html>
<html lang="sk">
    <head>
        <meta charset="UTF-8">
        <title>Kalkulačka</title>
    </head>
    <body>
        <p>Vitajte v kalkulačke, zadajte 2 čísla a získajte ich súčet.</p>

        <form method="POST" action="sucet.php">
            <input name="cislo1" type="number" /><br />
            <input name="cislo2" type="number" /><br />
            <input type="submit" value="Sčítaj" />
        </form>

    </body>
</html>
```

Výsledek by měl vypadat takto:



Na HTML stránce máme formulář a v něm pár vstupních textových polí typu text. Pokud ctíte HTML 5, můžete použít jako typ polí ve formuláři "number". Poslední pole je odesílací tlačítko, které vyvolá odeslání formuláře.

Zajímavější jsou atributy formuláře. Method="POST" udává způsob, jakým se data z formuláře odešlou. Ačkoli POST není výchozí hodnota, nedává pro formuláře příliš smysl používat žádnou jinou. Pokud bychom atribut neuvedli, použila by se metoda GET, kde by formulář odeslal data jako query string do URL adresy. Tak by byly jednak vidět a také je délka adresy omezená. Používejte tedy vždy POST. Action označuje skript, který

formulář zpracuje. Pokud atribut neuvedeme, odešle se formulář do toho samého souboru, ve kterém se nachází.

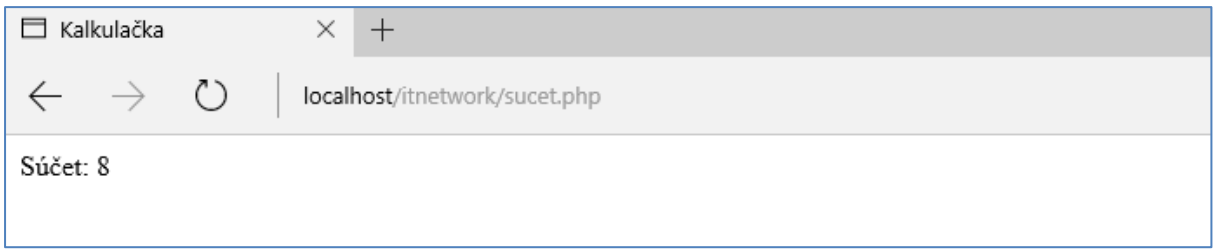
sucet.php

Soubor je skript s obslužným kódem, kterému se data z formuláře odešlou a on je zpracuje. Asi vás nepřekvapí, že data z formuláře přijdou v **superglobálním poli $_POST**. Opět je asociativní a klíčem jsou názvy textových (nebo jiných) polí, tedy jejich atributy name.

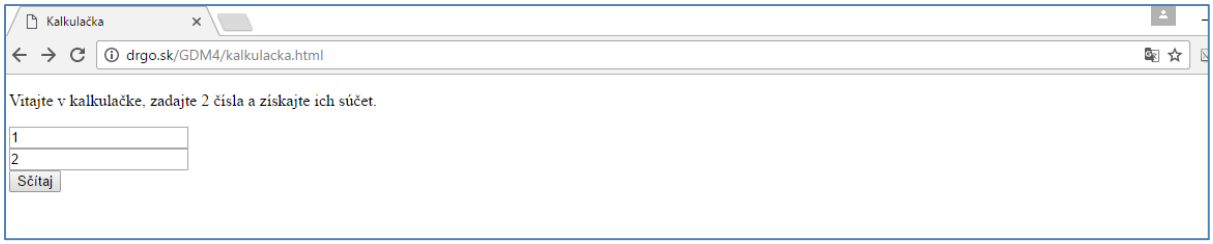
Obsah skriptu bude následující:

```
<!DOCTYPE html>
<html lang="sk">
    <head>
        <meta charset="UTF-8">
        <title>Kalkulačka</title>
    </head>
    <body>
        <?php
            $sucet = $_POST['cislo1'] + $_POST['cislo2'];
            echo("Súčet: $sucet");
        ?>
    </body>
</html>
```

Výstup:



Výstup z môjho PHP servera



Cvičení k 5. lekci PHP

Jednoduchý příklad

Vytvořte program, který si na vstupu nechá zadat jméno uživatele a poté jeho vlastnost. Nakonec vypíše "jméno je vlastnost", viz obrázek. K zadání použijte formulář, který bude odesílat data metodou POST.

Vlastnost.html

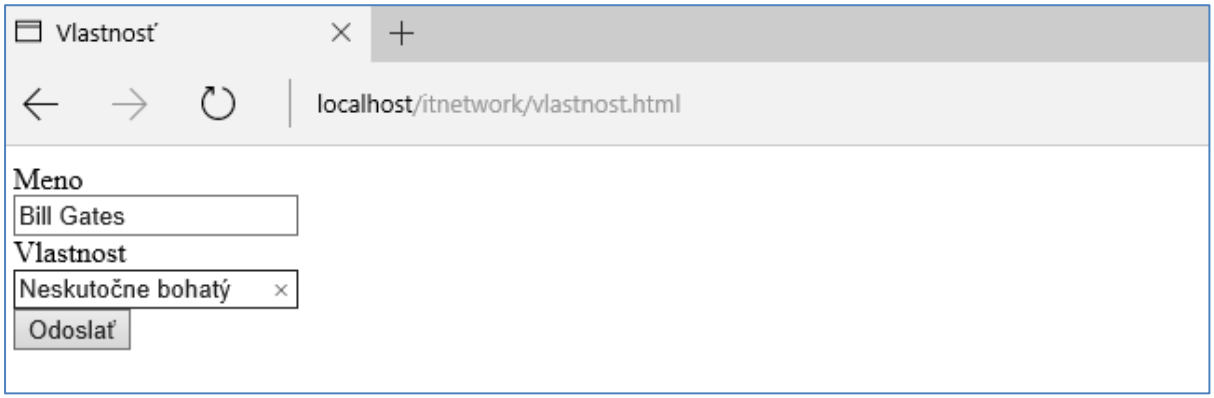
!!!Treba nastaviť prezeranie html súboru a nie php

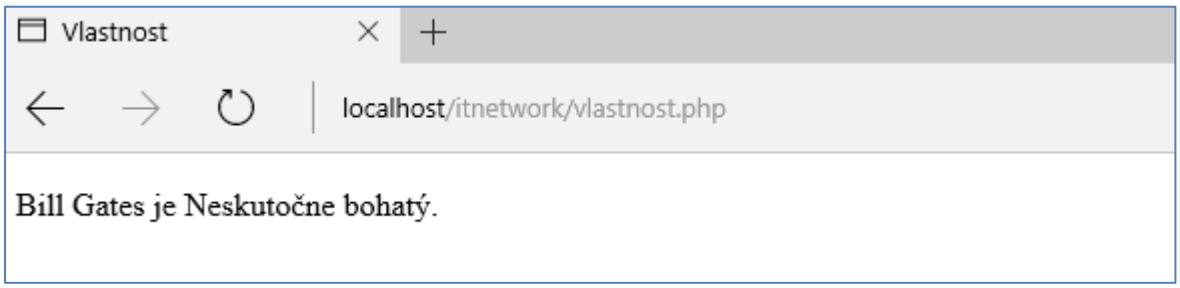
```
!DOCTYPE html>

<html lang="sk">
<head>
    <meta charset="utf-8" />
    <title>Vlastnosť</title>
</head>

<body>

    <form action="vlastnost.php" method="POST">
        Meno<br />
        <input type="text" name="meno" /><br />
        Vlastnost<br />
        <input type="text" name="vlastnost" /><br />
        <input type="submit" value="Odoslať" />
    </form>

</body>
</html>
```

vlastnost.php

```
<!DOCTYPE html>

<html lang="sk">
<head>
        <meta charset="utf-8" />
        <title>Vlastnost</title>
</head>

<body>
        <p>

                <?php


                echo($_POST['meno'] . ' je ' . $_POST['vlastnost'] . '.');
                ?>

        </p>
</body>
</html>
```

Ukázka obrazovky programu:





6. díl - Podmienky v PHP

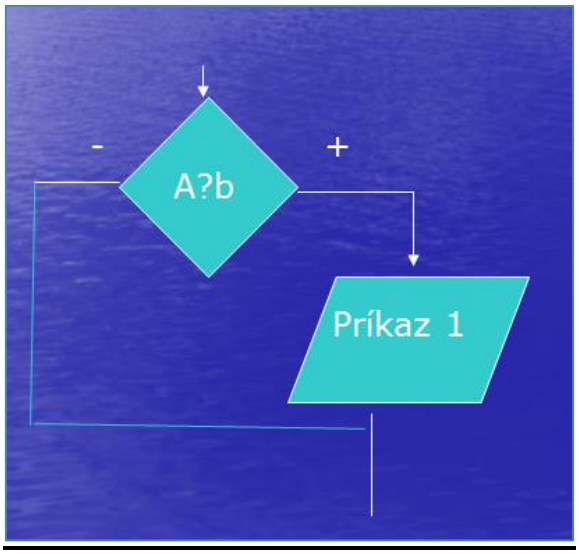
minulém dílu seriálu tutoriálů se základy PHP jsme si ukázali jak fungují formuláře a vytvořili si jednoduchou kalkulačku. Ta uměla jen sčítat. Dnes ji rozšíříme o další funkce. Budeme k tomu potřebovat podmínky.

Podmienky-vetvenia

Podmienky (alebo ak programátorské vetvenia) umožňujú, aby sa skript nesprával stále rovnako, ale reagoval na rôzne situácie. Najčastejšie reagujeme na vstup od užívateľa alebo rôzne udalosti.

1.Jednoduché vetvenie

Grafický zápis



Vetvenie v PHP

```
if ($a < $b){
        príkaz1;
        }
Príkaz 2;
```

Podmienku zapíšeme pomocou kľúčového slova if, za ktorým nasleduje v zátvorke logický výraz. Ak je výraz pravdivý, vykoná sa nasledujúci príkaz. Ak nie, nasledujúci príkaz vykonaný nebude a program pokračuje až pod ním.

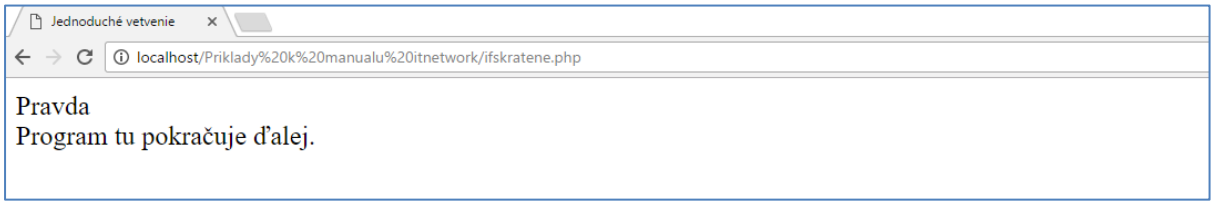
KOSTRA PRIKLADU:

```
if (15 > 5){
        echo('Pravda');
          }
echo('<br />Program zde pokračuje dál.');
```

Priklad ifskratene.php

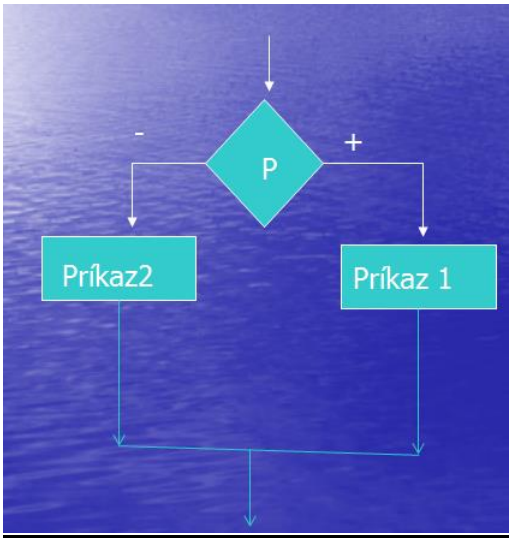
```
<DOCTYPE html>

<html lang="sk">
<head>
    <meta charset="utf-8" />
    <title>Jednoduché vetvenie</title>
</head>

<body>
        <?php

         if (15 > 5){
                        echo('Pravda');
                    }
        echo('<br />Program tu pokračuje ďalej.');

        ?>
</body>
</html>
```

Výstup programu nyní bude:



2.Uplný tvar podmienky if

Grafický zápis



SYNTAX:

```
if (podmienka){
                Príkaz1;
                }
 else{
        Príkaz2;
        }
```

Priklad 6.1 ifuplne.php

```
<?php
// Abecedné Porovnanie
  $a = "brian";
  $b = "zebra";
      if ($a < $b){
               echo $a." je pred ".$b." v abecede";
                       }
      else{
          echo $a." je po ".$b." v abecede";
      }
// Výsledok-Result : brian je pred zebra v abecede
?>
```

Operátory

V podmienke sme použili operátor <(menší).

Vo výrazoch môžeme ďalej používať tieto relačné operátory:

Operátor	zápis
Rovnosť	==
Je väčší	>
Je menší	<
Je väčší alebo rovný	>=
Je menší alebo rovný	<=
nerovnosť	!=
Všeobecná negácia	!

Rovnosť zapisujeme dvoma == preto, aby sa to neplietlo s bežným priradením do premennej, ktoré sa robí len jedným = . Ak chceme nejaký výraz znegovat, napíšeme ho do zátvorky a pred neho výkričník. Keď budeme chcieť vykonať viac než len jeden príkaz, musíme príkazy vložiť do bloku zo zložených zátvoriek. Súčasťou výrazu samozrejme môžu byť aj premenné.

Urobme si príklad na použitie operátorov nerovnosť a rovnosť

Priklad vetvenie1.php

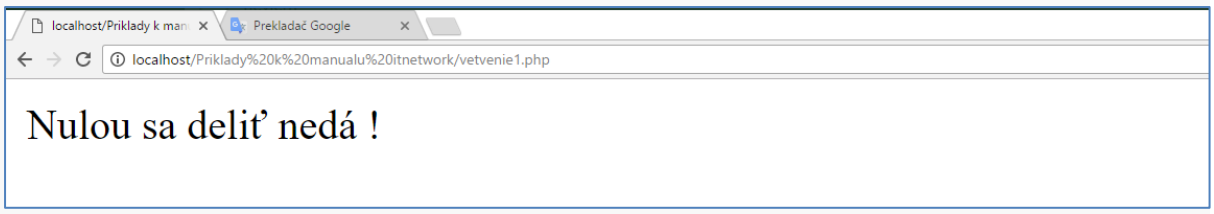
Kostra prikladu

```
$a = 10;
$b = 0;
if ($b != 0)
        {
         $vysledek = $a / $b;
         echo('Podiel: ' . $výsledok);
        }
if ($b == 0)
         echo('Nulou sa nedá deliť !');
```

Celé riešenie vetvenie1.php

```php
<?php

        $a = 10;
        $b = 0;
        if ($b != 0)
                    {
                        $vysledok = $a / $b;
                        echo('Podiel: ' . $vysledok);
                    }
        if ($b == 0)
                echo('Nulou sa deliť nedá !');

        ?>
```

Výstup



Skript vyššie spočíta a vypíše podiel dvoch premenných. Ak je druhá premenná nulová, vypíše chybu, pretože nulou sa samozrejme deliť nedá. Ak by premenné pochádzali od užívateľa a my by sme ich podmienkou neošetrili, mohol by nám takýto používateľ aplikáciu aj rozbiť. S ošetrením užívateľských vstupov úzko súvisí bezpečnosť webových aplikácií a počas seriálu uvidíte, ako moc je to dôležité.

Použitie else vo vetvení

Podmienku sme vyššie napísali vlastne dvakrát, raz normálne a raz znegovanou. Keby bola zložitejšia, mohli by sme v jej negáciu urobiť chybu. Ak chceme, aby sa niečo urobilo v prípade, že podmienka platí a niečo iné v prípade, že podmienka neplatí, použijeme kľúčové slovo else. Do vetvy else program zabehne v prípade, že podmienka neplatí.

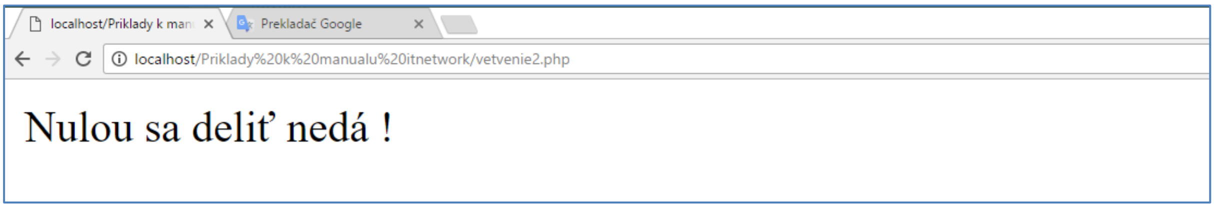
Kostra príkladu

```
$a = 10;
$b = 0;
if ($b != 0)
        {
         $vysledok = $a / $b;
         echo('Podiel: ' . $vysledok);
        }
else
         echo('Nulou sa deliť nedá!');
```

vetvenie2.php

```
<?php
        $a = 10;
        $b = 0;
            if ($b != 0)
                    {
                        $vysledok = $a / $b;
                        echo('Podiel: ' . $vysledok);
                    }
            else
                echo('Nulou sa deliť nedá !');
    ?>
```

Výstup scriptu



Ak by vo vetve **else** bolo viac príkazov, opäť by museli byť obsiahnuté v bloku zo zložených zátvoriek.

IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIITU POKRAČOVATIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII

CVIČENIE –VYTVORENIE KALKULAČKY

Potrebujeme 2 súbory:

- kalkulacka.html
- vypocet.php

Začnime HTML formulárom, do ktorého pridáme voľbu pre zvolenie početné operácie. Pridáme do neho tag **select,** ktorým zvolíme akú početnej operáciu chceme vykonať. Pre úplnosť si uveďme celý HTML súbor:

kalkulacka.html

```
<!DOCTYPE html>
<html lang="sk">
    <head>
        <meta charset="UTF-8">
        <title>Kalkulačka</title>
    </head>
    <body>
      <h1>Vitajte v  programe kalkulačka, zadajte 2 čísla a operaciu.</h1>

        <form method="POST" action="vypocet.php">
            Čísla:<br /><br />
            <input name="cislo1" type="number" /><br />
            <input name="cislo2" type="number" /><br /><br />
            Operácia:<br /><br />
            <select name="operacie"><br />
               <option value="scitanie">Sčítánie</option>
               <option value="odcitanie">Odčítanie</option>
               <option value="nasobenie">Násobenie</option>
               <option value="delenie">Delenie</option>
            </select><br /><br/>
            <input type="submit" value="Vypočítaj" />
        </form>
    </body>
</html>
```

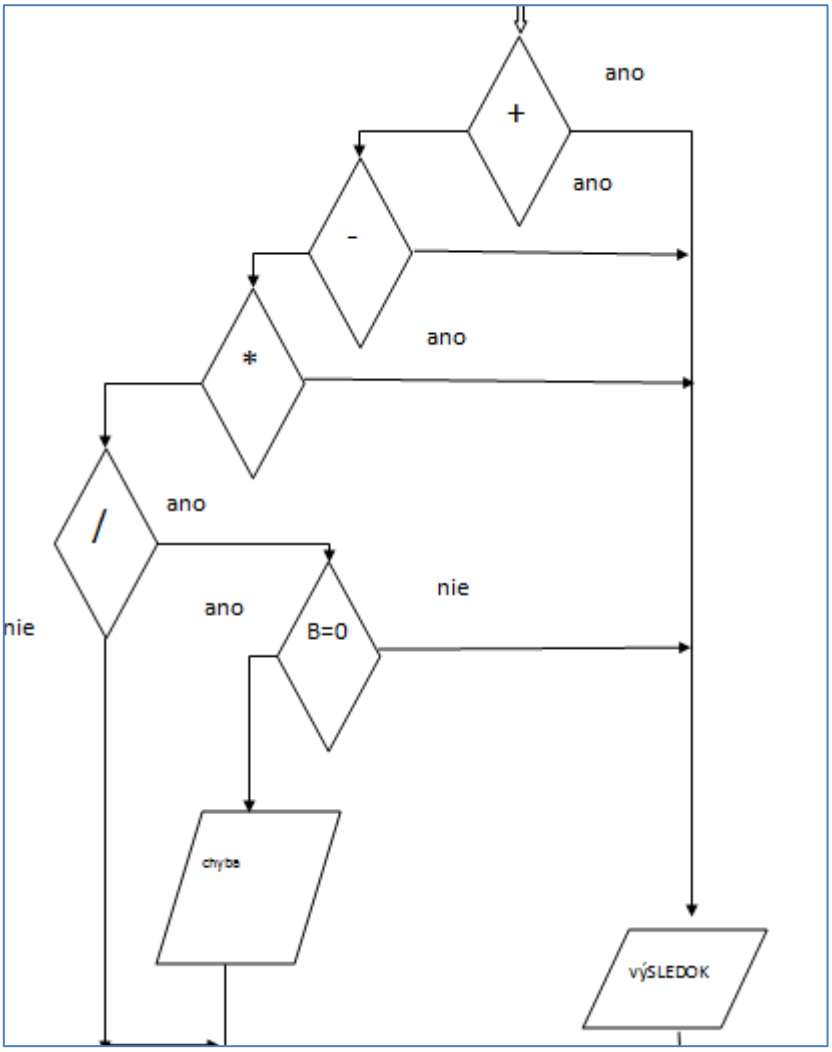
Obsluha voľby operácie.Tvorba súboru vypocet.php

Až sa formulár odošle, budeme mať v $ _POST ['operácie'] hodnotu podľa toho, akú možnosť užívateľ vybral.Podme robiť obslužný skript, ktorý sa volá vypocet.php. Dovnútra vetvenia dodajme pred delením nulou. Najprv si pre prehľadnosť na začiatku skriptu načítame vstupy z POSTu:

Kostra pre načítanie hodnoty čísel:

```
$a = $_POST['cislo1'];

$b = $_POST['cislo2'];

$operacie = $_POST['operacie'];
```

Princip scriptu:



Ďalej vložíme vetvenia na výpočet podľa typu operácie. Keďže je zbytočné, aby sa v overovaní obsahu premennej **operácie** pokračovalo aj v pripade, že sa neaká operácia uskutočnila, použijeme sekvenciu if ... else if ...(else znamená inak).

Kostra scriptu:

```
if ($operacie == 'scitanie')
        $vysledok = $a + $b;
```

```
else if ($operacie == 'odcitanie')
        $vysledok = $a - $b;
else if ($operacie == 'nasobenie')
        $vysledok = $a * $b;
else if ($operacie == 'delenie')
       {
        if ($b != 0)
                $vysledok = $a / $b;
        else
                $vysledok = 'Chyba';
       }
echo("Výsledok: $vysledok");
```

Poznámka:

Niekedy sa môžete stretnúť s tým, že sa **else if** spoja do jedného slova (**elseif**).

Ak je operácia napr. sčítanie, ďalšie **else** sa už nevyhodnotí, pretože podmienka platila a prejde sa rovno k vypísanie výsledku. Je teda vždy vykonané len toľko podmienok, koľko je potrebné.

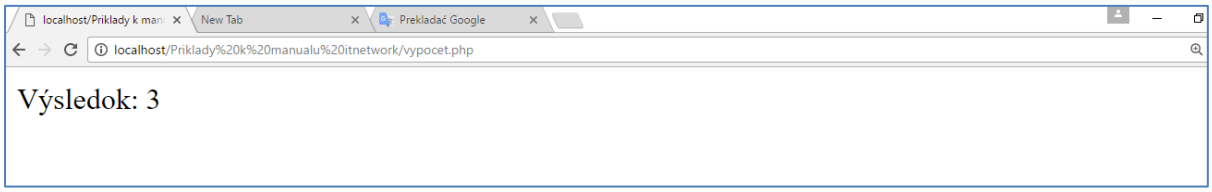
vypocet.php-výkonný script

```
<?php
                        $a = $_POST['cislo1'];
                        $b = $_POST['cislo2'];
                        $operacie = $_POST['operacie'];

                        if ($operacie == 'scitanie')
                                $vysledok = $a + $b;
                        else if ($operacie == 'odcitanie')
                                $vysledok = $a - $b;
                        else if ($operacie == 'nasobenie')
                                $vysledek = $a * $b;
                        else if ($operacie == 'delenie')
                        {
                                if ($b != 0)
                                        $vysledok = $a / $b;
                                else
                                $vysledok = 'Chyba,nulov deliť nevieme !!!';
                        }
            echo("Výsledok: $vysledok");
        ?>
```

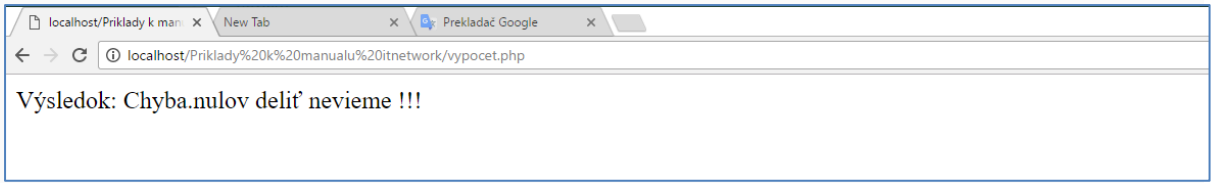
Výstup:

Formular:

PHP:



PHP: v prípade ,že druhé číslo bude 0



Cvičení k 6. lekci PHP

Jednoduchý príklad

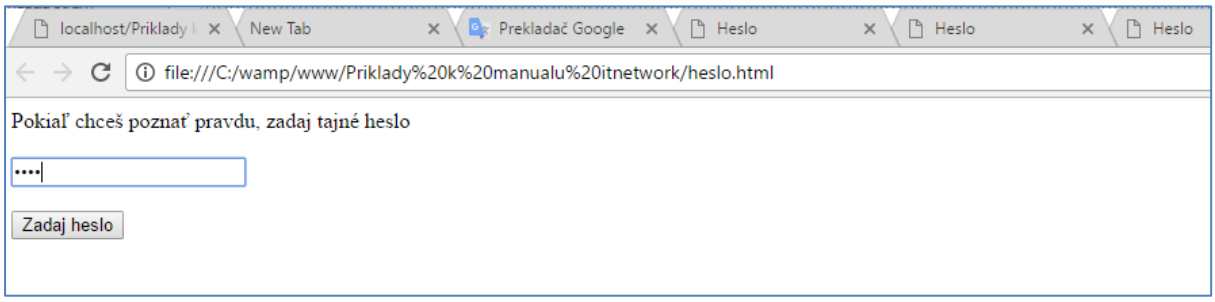
Vytvorte formulár pre zadanie hesla, v ktorom využite input typu **password**. Po odoslaní formulára programu overí, či bolo zadané heslo "veslo". Ak áno, vypíše tajný text "Urobil to knihovník." V opačnom prípade užívateľovi oznámi, že je jeho heslo neplatné.

Priklad heslo.html

```
<!DOCTYPE html>
<html lang="sk">
<head>
```

```
    <meta charset = "utf-8" />
    <title>Heslo</title>
</head>

<body>
    <form action="heslo.php" method="post">
        Pokiaľ chceš poznať  pravdu, zadaj tajné heslo<br /><br />
        <input type="password" name="heslo" /><br /><br />
        <input type="submit" value="Zadaj heslo" />
    </form>
</body>
```

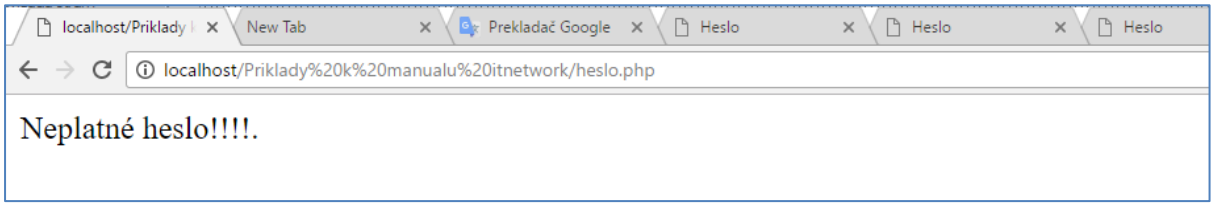
Formular:



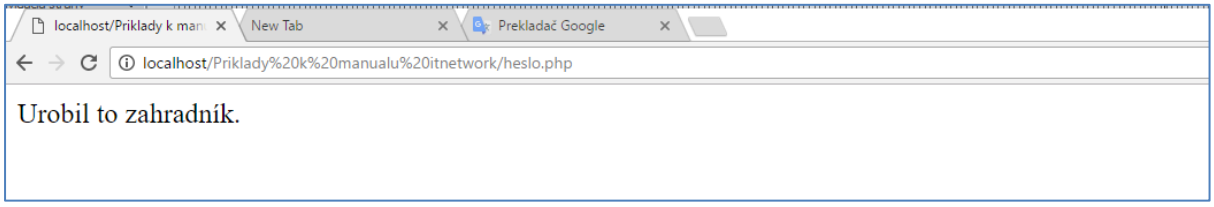
Súbor heslo.php

```
<?php


                if ($_POST['heslo'] == 'veslo')
                {
                        echo('<p>Urobil to zahradník.</p>');
                }
                else
                {
                        echo('<p>Neplatné heslo!!!!.</p>');
                }

        ?>
```

Výstup1:



Výstup2:



Stredne pokročilý príklad

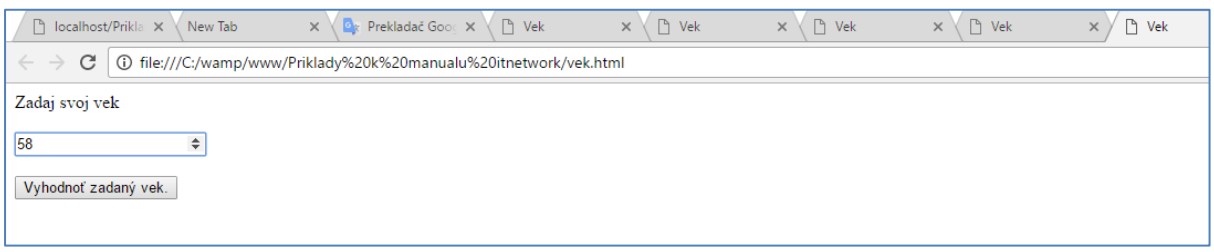
Vytvorte aplikáciu, ktorá zhodnotí vek používateľa. Vek si nechá zadať do formulára a po jeho odoslaní vypíše v závislosti na veku:

- 1-5 rokov: Si v predškolskom veku.
- 6-17 rokov: Ešte nie si dospelý.
- 18-49 rokov: Si mladý!
- 50+: Si už starý

Priklad vek.html

```
<!DOCTYPE html>
<html lang="sk">
<head>
    <meta charset="utf-8" />
    <title>Vek</title>
</head>

<body>
    <form action="vek.php" method="post">
        Zadaj svoj vek<br /><br />
        <input type="number" name="vek" /><br /><br />
        <input type="submit" value="Vyhodnoť zadaný vek." />
    </form>
</body>

</html>
```

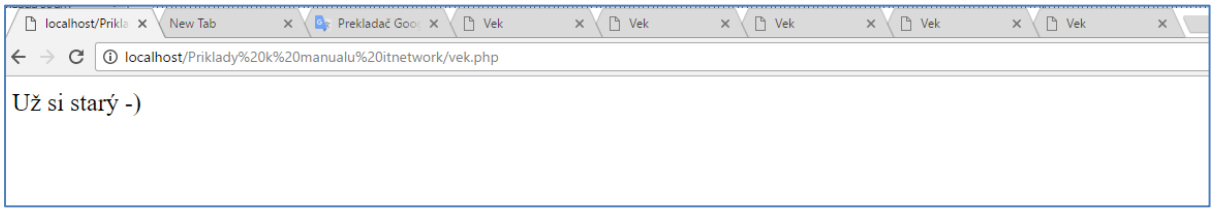
Formular:



Súbor vek.php

```php
<?php

		$vek = $_POST['vek'];
		if ($vek < 6)
		{
			echo('Si ešte  v preškolskom veku.');
		}
		else if ($vek < 18)
		{
			echo('Ešte nie si dospelý.');
		}
		else if ($vek < 50)
		{
			echo('Si ešte mladý!');
		}
		else
		{
			echo('Už si starý -)');
		}
	?>
```

PHP:



Tu treba pokračovať.operiť správnosť scriptu

Pokročilý příklad

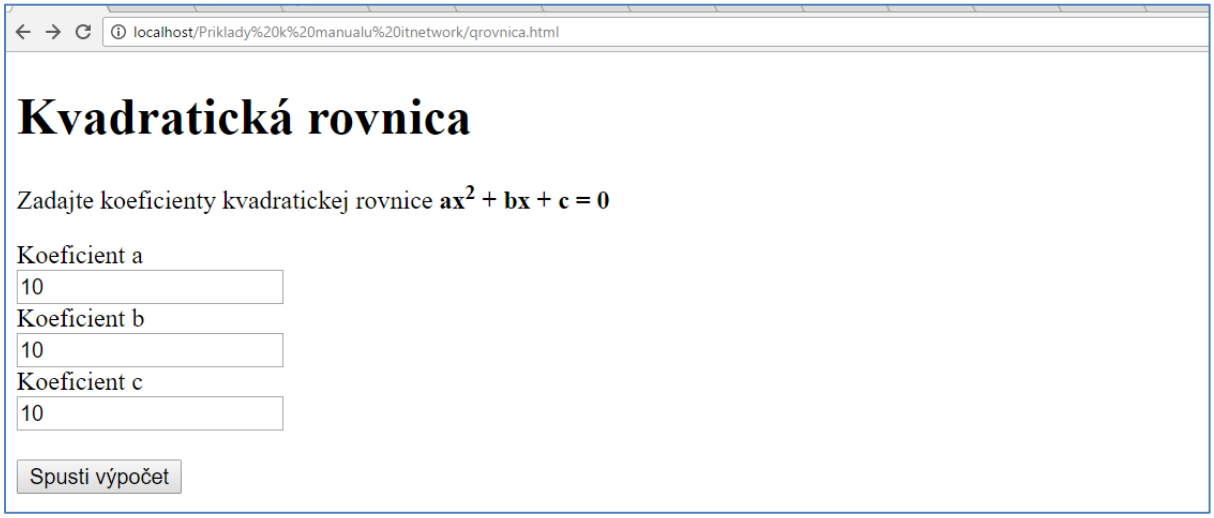
Vytvorte program, ktorý si na vstupe vyžiada postupne koeficienty a, b, c kvadratickej rovnice ax2 + bx + c = 0 a vypočíta jej reálne korene pomocou diskriminantu. Komplexnými korene sa nezaoberajte, pri zápornom diskriminante teda program vypíše, že rovnica nemá riešenie.Odmocninu zistíte pomocou **funkcie sqrt.**

K zadanie koeficientov použite formulár.

Priklad qrovnica.html

```
<!DOCTYPE html>
<html lang="sk">
<head>
    <meta charset="utf-8" />
    <title>Kvadratická rovnica</title>
</head>

<body>
<form action="qrovnica.php" method="post">
    <h1>Kvadratická rovnica</h1>
    <p>Zadajte koeficienty kvadratickej rovnice <strong>ax<sup>2</sup> + bx
+ c = 0</strong></p>
    Koeficient a<br />
    <input type="number" name="a" /><br />
   Koeficient b<br />
    <input type="number" name="b" /><br />
    Koeficient c<br />
    <input type="number" name="c" /><br /><br />
    <input type="submit" value="Spusti výpočet" />
</form>
</body>

</html>
```

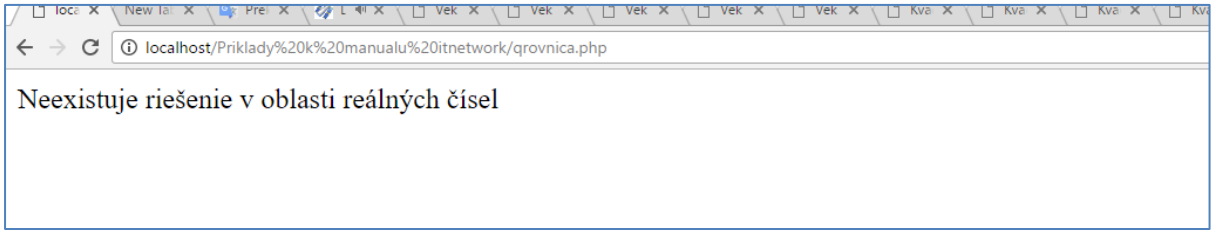
formular



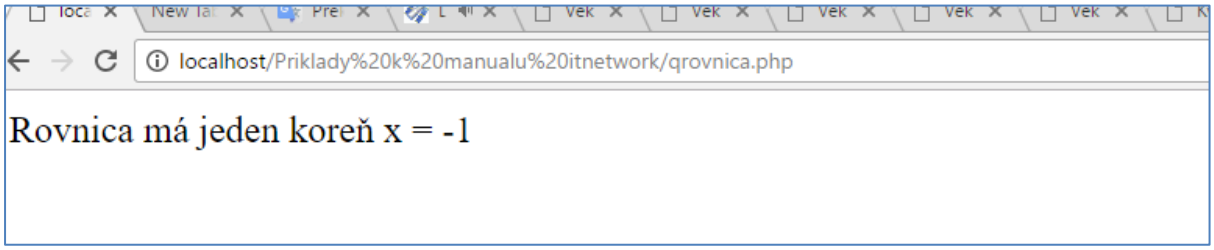
Príklad qrovnica.php

```php
<?php

                $a = $_POST['a'];
                $b = $_POST['b'];
                $c = $_POST['c'];
                if ($a != 0)
                {
                        // výpočet diskriminantu
                        $d = $b * $b - 4 * $a * $c;
                        if ($d < 0)
                                echo("<p>Neexistuje riešenie v  oblasti
reálnych čísel</p>");
                                // Pre zjednodušenie sa komplexnými korňmi
nebudeme zaoberať
                        else if ($d == 0)
                        {
                                $x = -$b / (2 * $a);
                                echo("<p>Rovnica má jeden koreň x =
$x</p>");
                        }
                        else
                        {
                                $x1 = (-$b + sqrt($d)) / (2 * $a);
                                $x2 = (-$b - sqrt($d)) / (2 * $a);
                                echo("<p>Rovnica má 2 reálne korene <br />x1
= $x1<br />x2 = $x2</p>");
                        }
                }
                else
                        echo("<p>Zadaná rovnica nie je kvadratická.</p>");

        ?>
```

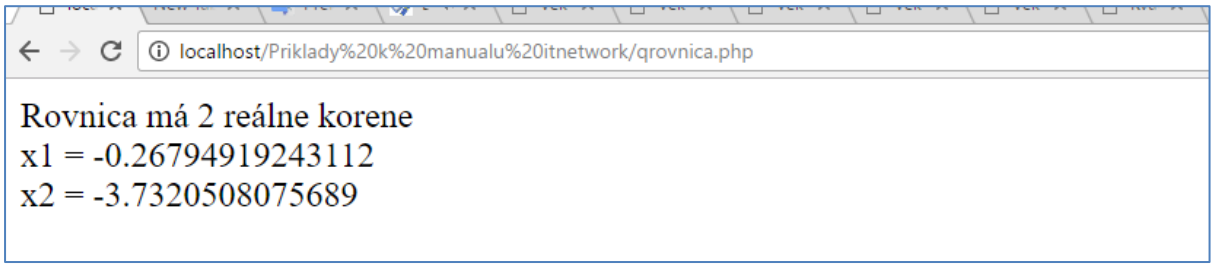
Výstup 10,10,10



Výstup 2,4,2



výstup:1,4,1



7. díl - Podmínky v PHP podruhé - přetypování, skládání a switch

V minulém dílu seriálu tutoriálů se základy PHP jsme si představili konstrukci if a else. Také jsem si ukázali relační operátory a rozšířili naši kalkulačku. Okolo podmínek je ještě nějaká teorie, která by vám do budoucna chyběla. Nezbývá, než se do ní pustit.

Datový typ boolean

S podmínkami úzce souvisí datový typ boolean. Ten nabývá pouze dvou hodnot a to true (pravda) a false (nepravda). Někdy bývají tyto konstanty psány velkými písmeny. Zkusme si takovou proměnnou vytvořit:

```
$pozdravit = true;
if ($pozdravit == true)
        echo('Ahoj');
```

Kód výše se ptá, zda je v proměnné $pozdravit hodnota true. Pokud ano, vypíše pozdrav. Hodnoty true a false se opravdu často používají u nějakého nastavení programu.

Když se na relační operátory v podmínkách zaměříme podrobněji, tak vždy vracejí výsledek typu boolean (např. 5 > 2 vrátí hodnotu true). Podmínka vždy skončí buď u toho, že platí nebo že neplatí. Do podmínky můžeme typ boolean vložit rovnou bez operátoru:

```
$pozdravit = true;
if ($pozdravit)
        echo('Ahoj');
```

Pokud výraz v závorce podmínky vrátí ve výsledku true nebo false, možná vás napadlo, zda jde výsledek podmínky uložit do proměnné typu boolean a použít ho k vyhodnocení až později. Ano, je to možné:

```
$podminka = (15 > 3);
if ($podminka)
{
        // ...
}
```

Porovnávání a přetypování

Již víme, že operátor == slouží k porovnání hodnot. A také víme, že je PHP dynamickým jazykem a datové typy proměnných samo převádí podle toho, jak s nimi pracujeme. K tomuto **přetypování** dochází i při psaní podmínek. Je opravdu velmi důležité, abyste věděli, jak funguje. Jinak vás chování PHP může překvapit nebo dokonce způsobíte bezpečnostní díru ve své aplikaci.

Přetypování na boolean

Pokud do podmínky zadáme proměnnou jakéhokoli datového typu, bude automaticky převedena na boolean (tedy na pravdu nebo nepravdu). Pokud jako výraz zadáme **nenulové číslo**, podmínka bude vždy platit a to i pro záporná čísla:

```
$a = 5;
if ($a)
        echo('Platí');
```

U řetězců bude podmínka platit v případě, že má nenulovou délku:

```
$s = 'devbook';
if ($s)
        echo('Platí');
```

U polí je to podobné. Pokud v poli něco je, bude podmínka platit:

```
$cisla = array(1, 2, 3);
if ($cisla)
        echo('Platí');
```

Toto může někdy zjednodušit práci. Podmínku s číslem využijeme v případě, že číslo označuje nějaký počet. Ušetříme si > 0. S polem je to podobné, tam si ušetříme count($cisla) > 0. Řetězec zas splní podmínku jen tehdy, když je zadaný.

U všech výše uvedených zápisů můžeme samozřejmě použít i negaci, např. takto:

```
if (!$jmeno)
        echo('Nemůžeš mít prázdné jméno.');
```

Porovnávání proměnných různých datových typů

Zajímavé to začne být zejména ve chvíli, kdy dojde při porovnávání k automatické (implicitní) konverzi datového typu. To se děje zejména u čísla a řetězce.

Asi vás nepřekvapí, že když zadáme to samé číslo jako číslo a jako řetězec, bude podmínka platit:

```
$a = 2;
$b = "2";
if ($a == $b)
        echo('Platí');
```

PHP zjistí, že je na levé straně číslo a pokusí se na číslo převést i stranu pravou. Zde se mu to podaří. Ve finále porovnává číslo 2 a číslo 2, které jsou stejné.

Možná si vzpomínáte, že PHP je poměrně aktivní (až moc) a snaží se převést typy za každou cenu. Podmínka v příkladu níže stále platí, i když by to už mohlo někoho zaskočit:

```
$a = 2;
$b = "2Ahoj, jak se máš?";
if ($a == $b)
    echo('Platí');
```

PHP vydoluje z řetězce dvojku a ve finále tedy porovnává zas dvě dvojky. Aby toto fungovalo, musí číslem řetězec začínat.

Do třetice si ukažme něco, co byste asi nečekali. Asi tušíte, že podmínka bude platit i v tomto posledním případě:

```
$a = 0;
$b = "Ahoj, jak se máš?";
```

```php
if ($a == $b)
    echo('Platí');
```

Proč tomu tak je? PHP zjistí, že porovnáváme číslo s něčím jiným a chce převést řetězec na číslo. Když se mu to nepodaří, místo chyby prohlásí, že je výsledek 0. A nula se přeci rovná nule 🙂

Porovnávání v závislosti na typu

Pokud vám výše zmíněné chování připadá poněkud divoké, máte pravdu. Někdy ušetří práci, ale někdy je lepší porovnávat v závisloti na datovém typu. To platí zejména pro textové řetězce. Pokud místou dvou == použijeme tři (===), výraz vrátí true jen v případě, že jsou hodnoty v proměnných opravdu stejné a mají stejný datový typ:

```php
$a = 0;
$b = "Ahoj, jak se máš?";
if ($a === $b)
    echo('Platí');
```

Nyní se již podmínka nesplní. To samé by platilo i pro číslo a řetězec:

```php
$a = 2;
$b = "2";
if ($a === $b)
    echo('Platí');
```

Textové řetězce bychom měli porovnávat spíše třemi ===.

Skládání podmínek

Podmínky je možné skládat a to pomocí dvou základních logických operátorů:

Operátor	C-like Zápis
A zároveň	&&
Nebo	||

Uveďme si příklad, zkontrolujme, zda je dané číslo v rozmezí 10-20:

```php
$a = 15;
if (($a >= 10) && ($a <= 20))
        echo("Zadal jsi správně");
else
```

```
        echo("Zadal jsi špatně");
```

Takto bychom mimochodem mohli validovat např. věk zadaný z formuláře. To již umíme. Operátory se pomocí závorek samozřejmě dají kombinovat. Zkontrolujme, zda je číslo v rozmezí 10-20 nebo 30-40:

Priklad zlozena_podmienka.html

Kostra prikladu

```
$a = 35;
if (((($a >= 10) && ($a <= 20)) || (($a >=30) && ($a <= 40)))
        echo("Zadal jsi správně");
else
        echo("Zadal jsi špatně");
```

Riešený príklad v slovenčine

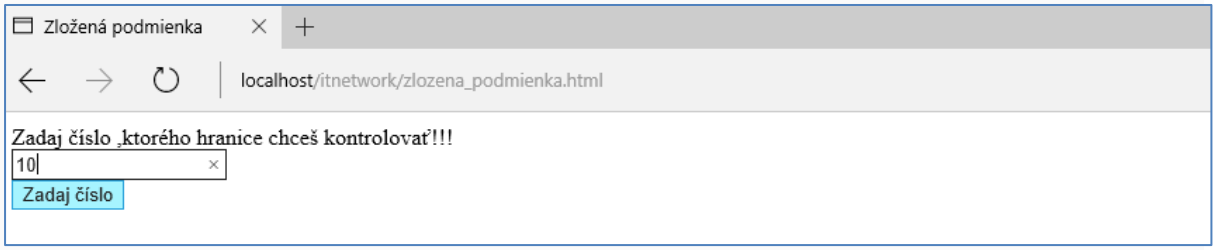
```
<!DOCTYPE html>
<html lang="sk">
<head>
    <meta charset = "utf-8" />
    <title>Zložená podmienka</title>
</head>

<body>
    <form action="zlozena_podmienka.php" method="post">
        Zadaj číslo ,ktorého hranice chceš kontrolovať!!!<br />
        <input type="number" name="cislo" /><br />
        <input type="submit" value="Zadaj číslo" />
    </form>
</body>

</html>
```

Súbor zlozena_podmienka.php

```
<!DOCTYPE html>
<html lang="sk">
<head>
    <meta charset = "utf-8" />
    <title>Zložená podmienka and alebo</title>
</head>

<body>
    <?php
    if (((($cislo >= 10) && ($cislo <= 20)) || (($cislo >=30) && ($cislo <=
40)))
        echo("Zadal si správne");
```

```
else
        echo("Zadal si zle!!!");
    ?>
</body>

</html>
```

Výstup



Switch

Abychom podmínky dobrali, představme si ještě konstrukci switch. Jedná se vlastně o alternativní zápis sekvence if...else. Switch (česky přepínač) umožňuje jednoduše reagovat na různé hodnoty nějaké proměnné. Ukažme si náš skript pro zpracování výpočtu kalkulačky, kde místo sekvence if...else použijeme právě switch:

Prepinac.html

Kostra programu

```
switch ($operace)
{
        case 'scitanie':
                $vysledek = $a + $b;
                break;
        case 'odcitanie':
                $vysledek = $a - $b;
```

```
                break;
        case 'nasobenie':
                $vysledek = $a * $b;
                break;
        case 'delenie':
                if ($b != 0)


                else
                        $vysledek = 'Chyba';
                break;
}

echo("Výsledek: $vysledek");
```

slovenska verzia programu Prepinac.html

```
<!DOCTYPE html>
<html lang="sk">
    <head>
        <meta charset="UTF-8">
        <title>Kalkulačka2</title>
    </head>
    <body>
        <p>Vitajte v kalkulačke22, zadajte 2 čísla a operaciu.</p>

        <form method="POST" action="prepinac.php">
            <input name="a" type="number" /><br />
            <input name="b" type="number" /><br />
            Operácia:
            <select name="operacie">
               <option value="scitanie">Sčítání</option>
               <option value="odcitanie">Odčítání</option>
               <option value="nasobenie">Násobení</option>
               <option value="delenie">Dělení</option>
            </select><br />
            <input type="submit" value="Vypočítaj" />
        </form>

    </body>
</html>
```

Subor prepinac.php

```
<!DOCTYPE html>
```

```
<html lang="sk">
<head>
    <meta charset = "utf-8" />
    <title>Demonštrácia prepínača</title>
</head>

<body>
    <?php
     $a = $_POST['a'];
     $b = $_POST['b'];
     $operacie = $_POST['operacie'];

    switch ($operacie)
{
        case 'scitanie':
                $vysledok = $a + $b;
                break;
        case 'odcitanie':
                $vysledok = $a - $b;
                break;
        case 'nasobenie':
                $vysledok = $a * $b;
                break;
        case 'delenie':

}

         echo("Výsledok: $vysledok");


    ?>
</body>

</html>
```

Za klíčovým slovem switch vložíme do závorky proměnnou, jejíž hodnoty chceme opodmínkovat. Tělo switche je v bloku ze složených závorek. Jednotlivé případy označíme slovem case a za ně vložíme dvojtečku. Dále normálně píšeme příkazy, které se mají stát. Nejsou zde žádné složené závorky a příkazů zde může být více. Za posledním je nutné vždy uvést příkaz break. Můžeme také použít případ default:, který se provede tehdy, když neplatí ani jedna možnost.

Osobně switch příliš nepoužívám. Nepřijde mi, že něco zjednodušuje a správný programátor ho příliš často nepotřebuje. Ono se takové bezduché větvení v programech většinou nepoužívá, jelikož jsou elegantnější a mnohem kratší zápisy, jak danou situaci řešit (třeba polymorfismem, reflexí nebo anonymními funkcemi, ale na to máme ještě hodně času). Určitě na switch ale narazíte v cizích programech a dobrý seriál by ho měl zmínit. Příště budeme zas dělat něco praktického a to kontaktní emailový formulář.

8. díl - Kontaktní emailový formulář v PHP

minulém dílu seriálu tutoriálů se základy PHP jsme dokončili podmínky. Na dnešek máme slíbený kontaktní formulář, do kterého návštěvníci našich stránek napíší vzkaz a ten se nám odešle emailem. Jedná se o velmi užitečný webový doplněk, díky kterému nás mohou uživatelé našich stránek lépe kontaktovat.

HTML část

Jako vždy bude aplikace rozdělena na 2 části. V tomto případě ovšem budou obě v jednom souboru mailform.php. Je to z toho důvodu, abychom měli při zpracování dat z formuláře přístupný i formulář. Pokud uživatel zadá něco špatně, vypíšeme nad formulář chybovou hlášku.

HTML část bude tedy obsahovat formulář, který bude mít následující prvky:

- **Jméno** - Jméno návštěvníka (abychom věděli kdo nám píše)
- **Emailová adresa** - Emailová adresa návštěvníka (abychom mu mohli odpovědět)
- **Zpráva** - Zpráva od uživatele
- **Antispam** - Ochrana proti spamu

Kromě ochrany proti spamu asi není co vysvětlovat. Řekněme si tedy o spamu více.

Spam

Jakmile vložíte na internet nějakou stránku s formulářem, objeví se časem roboti, kteří do formuláře začnou psát reklamu. Důvod je prostý, formulář někam něco odesílá a když do něj vloží odkaz na nějaké služby (často půjčky nebo pornografii), část lidí na tu reklamu klikne a služby si koupí.

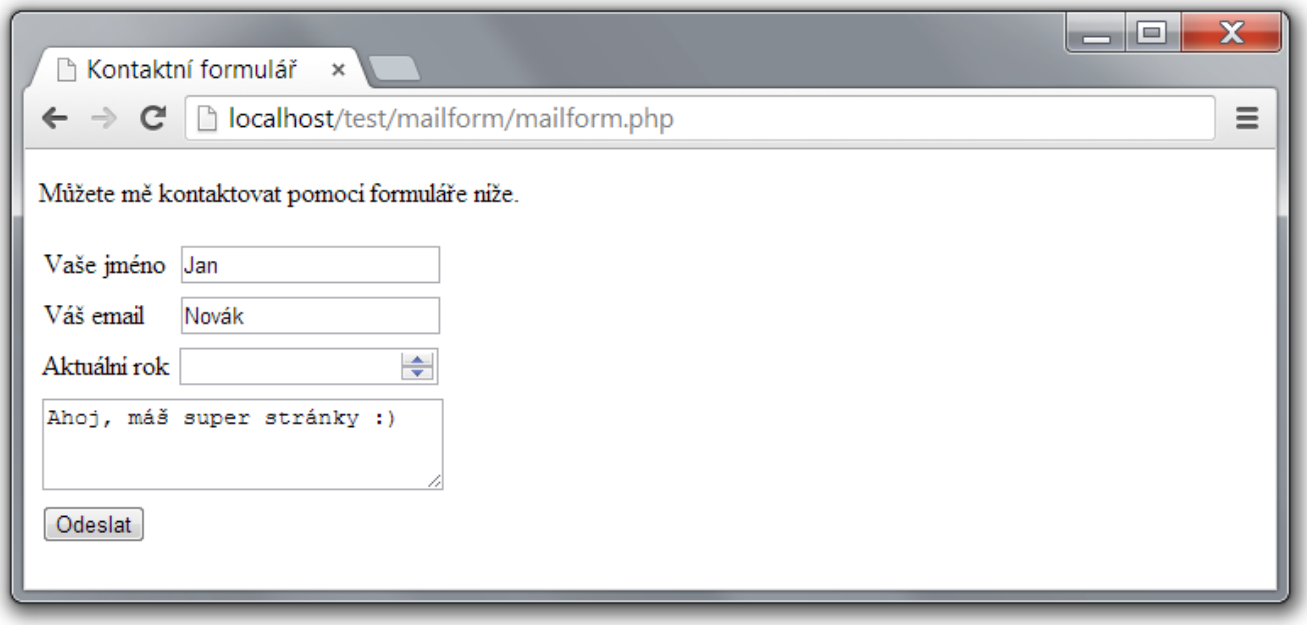
Proti spamu se dá velmi účinně bránit. K zabezpečení formulářů se používá tzv. Turingův test, známý spíše pod pojmem Captcha. Účelem testu je položit takovou otázku, na kterou zná odpověď jen člověk. První captchy často zobrazovaly text na obrázku a předpokládalo se, že obrázek umí přečíst jen člověk. Postupem času však spameři vyvinuli poměrně sofistikované OCR čtečky, které umí obrázky číst lépe, než lidé. Není ovšem nic jednoduššího, než položit nějakou otázku (nejlépe česky), kterou spamboti neumějí. Bohatě nám bude stačit např. "Zadejte aktuální rok".

Formulář

Založte si tedy nový projekt a můžeme začít. HTML kód stránky s formulářem by mohl vypadat např. takto:

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
        <title>Kontaktní formulář</title>
    </head>
    <body>
        <p>Můžete mě kontaktovat pomocí formuláře níže.</p>

        <form method="POST">
            <table>
                <tr>
                        <td>Vaše jméno</td>
                        <td><input name="jmeno" type="text" /></td>
                </tr>
                <tr>
                        <td>Váš email</td>
                        <td><input name="email" type="email" /></td>
                </tr>
                                <tr>
                        <td>Aktuální rok</td>
                        <td><input name="rok" type="number" /></td>
                </tr>
                </table>
                <textarea name="zprava"></textarea><br />

            <input type="submit" value="Odeslat" />
        </form>

    </body>
</html>
```

A výsledek:



Formulář jsme vložili do tabulky, aby byly prvky hezky zarovnané. Dělá se to tak často, pokud chceme formulář rychle nastylovat. Všimněte si, že ve formuláři není vyplněný parametr action. Data se tedy odešlou na ten samý soubor, ve kterém je formulář.

PHP část

Na úplný začátek souboru vložíme PHP direktivu a pustíme se do programování:

```
<?php

?>
```

Validace

Každý formulář bychom měli zvalidovat. Validace je ověření, zda je správně vyplněný. Sice ještě neumíme ověřit, zda jsou v polích správné hodnoty, ale umíme zjistit, zda nejsou prázdná.

Kromě toho, že pole přišlo prázdné, je ještě jedna možnost - formulář se nemusel vůbec odeslat. S touto možností musíme počítat, jelikož máme zpracování i zobrazení ve stejném skriptu - uživatel mohl zatím jen zobrazit formulář a nic neodeslat. Z minulé lekce víme, že pokud napíšeme:

```
if ($_POST)
```

Provede se podmínka v případě, když pole není prázdné.

Dále bychom měli počítat i s tím, že se formulář neodeslal celý, ale jen jeho část. Potřebujeme tedy zjistit, zda v $_POST existují jednotlivé proměnné. K tomu v PHP slouží funkce isset().

POZOR! Mnoho začátečníků používá k ověření toho, zda se něco odeslalo, následující kód:

```
if ($_POST['jmeno'])
{
        // ...
}
```

To je ovšem špatně a pokud se formulář neodeslal, PHP vypíše ošklivou chybu, jelikož čteme z neexistující proměnné. Tito začátečníci si místo toho, aby kód opravili, vypnou výpis chyb v PHP. Později sem chodí a diví se, že jim něco nefunguje a nemohou chyby najít.

Když už jsme u nastavení chyb, tak musí být vždy takové, že jsou na lokálním serveru (na vašem počítači při testování) zapnuté a na produkci (na internetu) vždy vypnuté. Jen tak odhalíte při testování většinu problémů a na produkci vám nikdo díky viditelné chybové hlášce nebude napadat aplikaci. Chyby lze zapínat a vypínat v php.ini, někdy ovšem na produkci nemusíme mít k tomuto nastavení přístup a existuje k tomu nějaký přepínač v administrátorském rozhraní daného webhostingu.

Validace formuláře by mohla vypadat asi takto:

```
$hlaska = '';
if ($_POST) // V poli _POST něco je, odeslal se formulář
{
        if (isset($_POST['jmeno']) && $_POST['jmeno'] &&
                isset($_POST['email']) && $_POST['email'] &&
                isset($_POST['zprava']) && $_POST['zprava'] &&
                isset($_POST['rok']) && $_POST['rok'] == date('Y'))
        {
                // Sem přijde odeslání emailu
        }
        else
                $hlaska = 'Formulář není správně vyplněný!';
}
```

Celý kód je v podmínce, která kontroluje, zda je něco v poli $_POST. Pokud se nic neodeslalo, není co zpracovávat. Další složená podmínka kontroluje, zda byla odeslána jednotlivá pole a zda v nich je nějaký text. U roku samozřejmě kontrolujeme, zda je aktuální. Ve skriptu používáme proměnnou $hlaska, kam vložíme hlášku pro uživatele v případě, že se validace nepovedla. Tu později vypíšeme v HTML části skriptu.

Zpracování

Samotné odeslání emailu není složité. Slouží k tomu funkce mb_send_mail(), která narozdíl od starší funkce mail() podporuje UTF-8 kódování. K funkcím s prefixem mb_ se ještě dostaneme, nyní nám musí stačit, že pokud je chceme používat, musíme na úplném začátku souboru nastavit kódování:

```
<?php
mb_internal_encoding("UTF-8");
```

Přejděme dovnitř naší podmínky s validací a umístěme tam odeslání emailu a nastavení zprávy pro uživatele:

```
$hlavicka = 'From:' . $_POST['email'];
$hlavicka .= "\nMIME-Version: 1.0\n";
$hlavicka .= "Content-Type: text/html; charset=\"utf-8\"\n";
$adresa = 'nas@email.cz';
$predmet = 'Nová zpráva z mailformu';
$uspech = mb_send_mail($adresa, $predmet, $_POST['zprava'], $hlavicka);
if ($uspech)
{
        $hlaska = 'Email byl úspěšně odeslán, brzy vám odpovíme.';
}
else
        $hlaska = 'Email se nepodařilo odeslat. Zkontrolujte adresu.';
```

Do několika proměnných si připravíme hlavičku, adresu, kam se má email odeslat (tu si samozřejmě nastavte na svou) a předmět. Jak vypadá hlavička je dané a nemusíte nad tím přemýšlet, podstatná je jen proměnná v prvním řádku, která určuje odesílatele emailu. Email potom vypadá jako že přišel z této adresy, i když ho odeslalo PHP z vašich stránek. Funkce mb_send_mail() vrací true pokud se odeslání podařilo a false pokud selhalo. Tuto hodnotu si uložíme do proměnné $uspech a nastavíme podle ní hlášku.

Úprava formuláře

Vraťme se k našemu formuláři a vložme těsně nad tag form další PHP sekvenci, ve které vypíšeme proměnnou $hlaska, pokud v ní něco je:

```
<?php
        if ($hlaska)
                echo('<p>' . $hlaska . '</p>');
?>
```

Hotovo. Váš formulář by měl nyní odesílat emaily a zobrazovat chybové hlášky. Musíte to ale vyzkoušet spíše tak, že si ho nahrajete někam na webhosting. V XAMPPu ve výchozím nastavení není odesílání emailů funkční, i když jdenastavit v konfiguračním

souboru. Pokud máte s nastavením problémy, nevadí, prostě si formulář někam nahrajte (třeba na webhosting OneBit) a vyzkoušejte ho online.

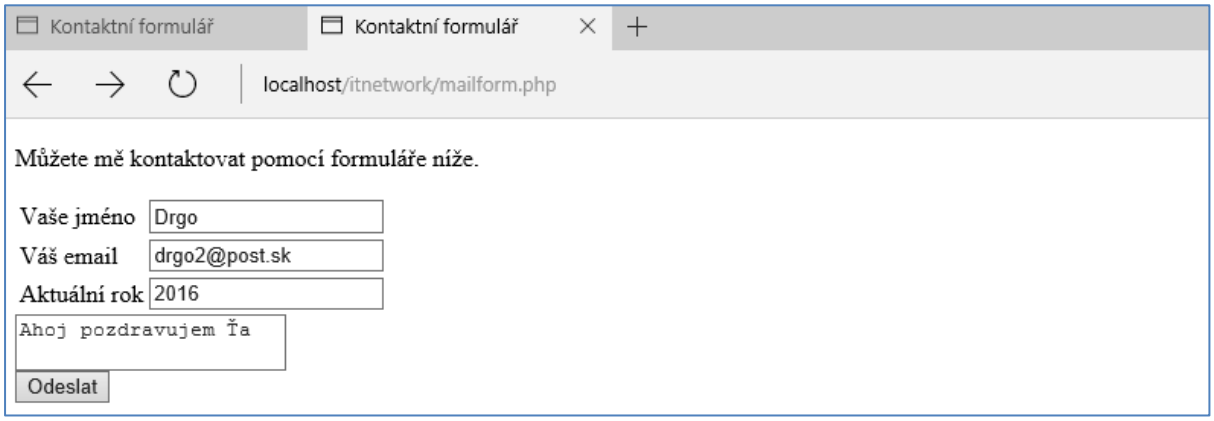
Priklad mailform.php

```
<?php

        /*
         *      Podľa itnetwork.cz
        */

    mb_internal_encoding("UTF-8");

    $hlaska = '';
    if ($_POST) // V poli _POST něco je, odeslal se formulář
    {
        if (isset($_POST['jmeno']) && $_POST['jmeno'] &&
                        isset($_POST['email']) && $_POST['email'] &&
                        isset($_POST['zprava']) && $_POST['zprava'] &&
                        isset($_POST['rok']) && $_POST['rok'] == date('Y'))
        {
            $hlavicka = 'From:' . $_POST['email'];
            $hlavicka .= "\nMIME-Version: 1.0\n";
            $hlavicka .= "Content-Type: text/html; charset=\"utf-8\"\n";
            $adresa = 'vas@email.cz';
            $predmet = 'Nová zpráva z mailformu';
            $uspech = mb_send_mail($adresa, $predmet, $_POST['zprava'],
$hlavicka);
            if ($uspech)
            {
                $hlaska = 'Email byl úspěšně odeslán, brzy vám odpovíme.';
            }
            else
                $hlaska = 'Email se nepodařilo odeslat. Zkontrolujte
adresu.';
        }
        else
            $hlaska = 'Formulář není správně vyplněný!';
    }

?>

<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Kontaktní formulář</title>
```

```
    </head>
    <body>
        <p>Můžete mě kontaktovat pomocí formuláře níže.</p>

        <?php
            if ($hlaska)
                echo('<p>' . $hlaska . '</p>');
        ?>

        <form method="POST">
            <table>
                <tr>
                        <td>Vaše jméno</td>
                        <td><input name="jmeno" type="text" /></td>
                </tr>
                <tr>
                        <td>Váš email</td>
                        <td><input name="email" type="email" /></td>
                </tr>
                        <tr>
                        <td>Aktuální rok</td>
                        <td><input name="rok" type="number" /></td>
                </tr>
                </table>
                <textarea name="zprava"></textarea><br />

            <input type="submit" value="Odeslat" />
        </form>

    </body>
</html
```

Příště formulář dokončíme. Pokud jste měli s nějakou částí problém, níže jsou zdrojové kódy ke stažení.

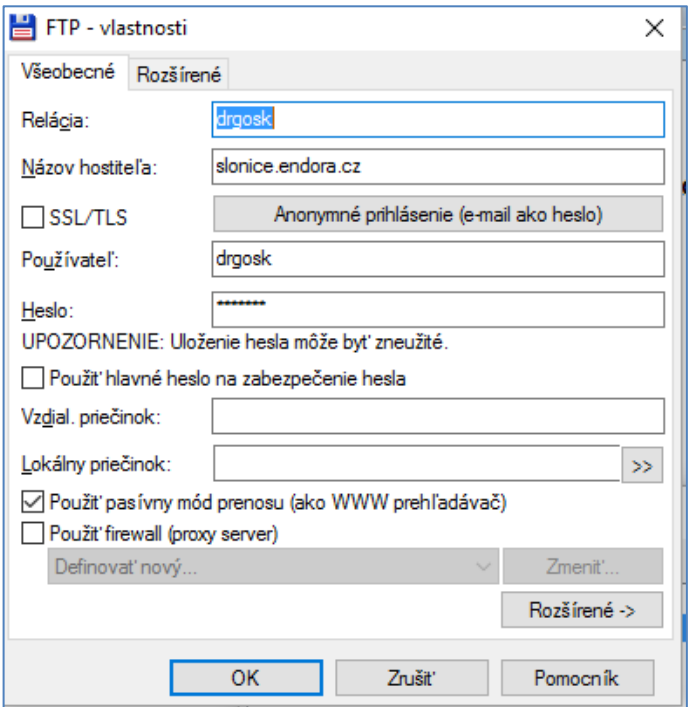
Výstup



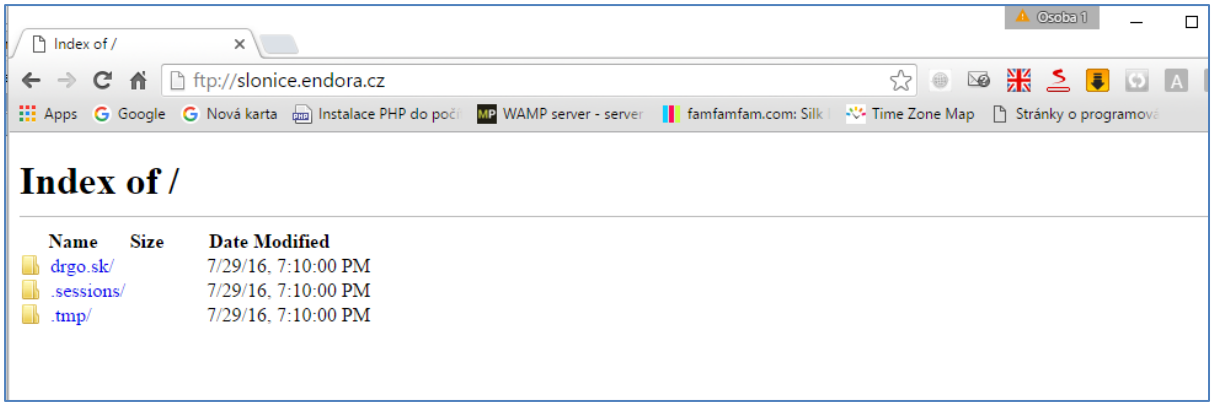
Tvorba www.drgo.sk

ftp://slonice.endora.cz

```
Doména: http://drgo.sk
Adresa administrace : http://webadmin.endora.cz
Uživatelské jméno: drgosk
Heslo: si zvolíte při aktivaci účtu
Kontaktní e-mailová adresa: drgo2@post.sk

Údaje FTP:
Hostitel: slonice.endora.cz
Uživatelské jméno: drgosk
Heslo: si zvolíte při aktivaci účtu
```



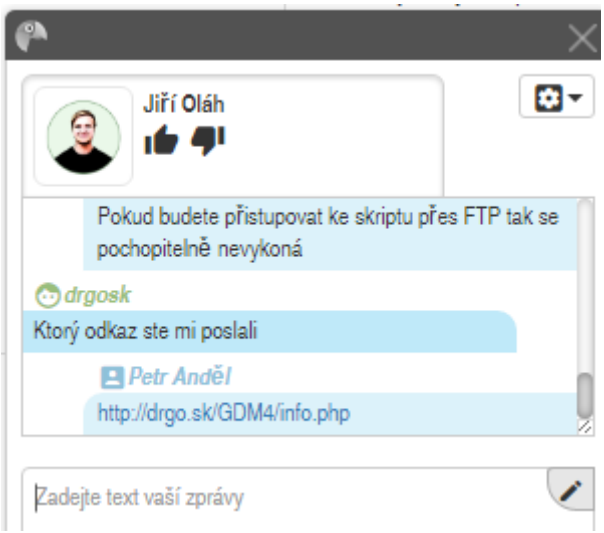
Prístup na web pomocou ftp

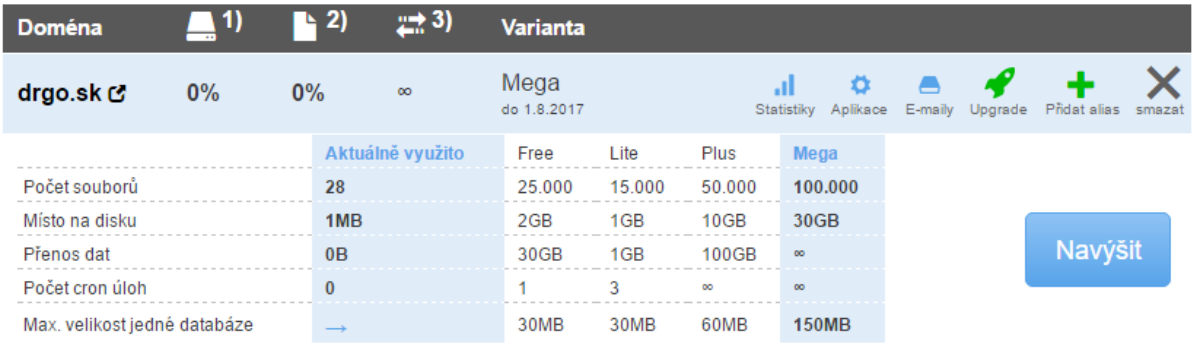


Pristup do priečinku GDM1

ftp://slonice.endora.cz/drgo.sk/web/GDM4/

Spúšťanie PHP scriptov so php servera:





Zaplatené do 1.8.2017

9. díl - Vylepšení kontaktního formuláře v PHP

V minulém dílu seriálu tutoriálů se základy PHP jsme naprogramovali jednoduchý emailový formulář. V dnešním dílu ho vylepšíme.

Předvyplnění polí

Když něco špatně vyplníme, skript nám to oznámí, ale formulář je již prázdný. Je to samozřejmě proto, že zobrazujeme již jinou stránku, než do které jsme data zadávali.

Protože zpracováváme data tím samým skriptem, ve kterém je formulář, můžeme do formuláře jednoduše vložit hodnoty z POSTu v případě, že byly nějaké odeslány. Budete to vypadat, jako by se formulář ani nevymazal.

V PHP bloku těsně před formulářem si naplníme proměnné $jmeno, $email a $zprava hodnotami z $_POST. To můžeme udělat samozřejmě jen v případě, když v POSTu tyto hodnoty jsou. Pokud ne, dáme do proměnných prázdné řetězce.

V této chvíli bychom kód napsali asi takto:

```
if (isset($_POST['jmeno']))
        $jmeno = $_POST['jmeno'];
else
        $jmeno = '';
```

Kód je poměrně dlouhý a psát ho pro všechny 3 proměnné by bylo nepohodlné. Zvláště, když by formulář obsahoval ještě další pole. Proto si uvedeme tzv. **ternární výraz**.

Jedná se o zkrácenou podobu if ... else. Ternání výraz **vždy vrací nějakou hodnotu**, nedá se tedy použít jen místo podmínky. Skládá se ze tří části, což by nás podle jeho názvu nemělo překvapit V první části uvedeme podmínku, dále znak ? a poté hodnotu, kterou má výraz vrátit, pokud podmínka platí. Za ní uvedeme : a hodnotu, která se má vrátit, když podmínka neplatí.

Proměnnou bychom pomocí ternárního výrazu naplnili takto:

```
$jmeno = (isset($_POST['jmeno'])) ? $_POST['jmeno'] : '';
```

Ještě malá rekapitulace. Pokud v POSTu existuje daný klíč, vložíme tuto hodnotu do proměnné $jmeno. Pokud ne, vložíme tam prázdný textový řetězec, což jsou jednoduše uvozovky, mezi kterými nic není.

Toto uděláme ještě s emailem a zprávou.

Formulář dále upravíme tak, aby se do jeho polí vkládaly hodnoty z těchto proměnných. Pokud nebyl formulář odeslán, vloží se tam prázdné řetězce, jinak se tam vloží to, co uživatel vyplnil.

Do formulářového pole bychom mohli hodnotu vložit takto:

```
<input name="jmeno" type="text" value="<?php echo $jmeno ?>" />
```

Pokud chceme v PHP sekvenci pouze vypsat obsah proměnné, použijeme k tomu zkrácenou direktivu <?=. Stejného výstupu tedy můžeme dosáhnout i kratším zápisem:

```
<input name="jmeno" type="text" value="<?= $jmeno ?>" />
```

Stále to však není ono. Jelikož **text v proměnné $jmeno pochází od uživatele**, nemůžeme si být jistí, že je bezpečný. Co kdyby nám do textu uživatel vložil nějaký HTML kód? Vložil by se poté normálně do naší stránky.

V našem případě by to tolik nevadilo, ale jsou případy, kde ano. Představte si, že vypisujete na stránce zprávy od uživatelů, třeba komentáře k nějakému článku. A jeden uživatel místo textu zprávy vložil HTML kód s formulářem, který je nasměrovaný na jeho obslužný skript na jiném webu a ve kterém požaduje od vašich uživatelů heslo. Při vypisování zprávy se tento kód vypíše do stránky a zobrazí se jeho formulář. Někteří vaši uživatelé tam opravdu zadají své heslo, které se odešle někam útočníkovi. A vy máte problém.

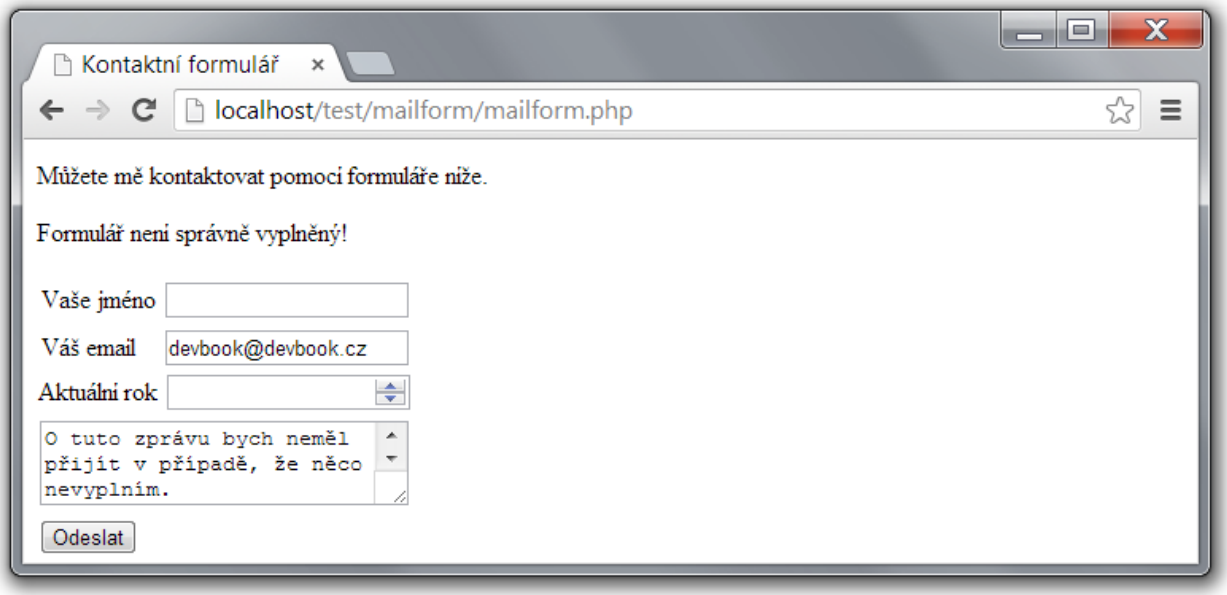
Tomuto typu útoku se říká XSS (jako Cross Site Scripting). Obrana je velmi jednoduchá, **před vypsáním obsahu kterékoli proměnné do HTML kódu stránky použijte funkci htmlspecialchars()**. Ta převede špičaté HTML závorky a pár dalších znaků na tzv. HTML entity. Případný škodlivý kód je potom braný jen jako obyčejný text a prohlížeč ho nezpracuje jako HTML kód.

Je potřeba při výpisu ošetřovat opravdu každou proměnnou, i ty, které uživatel přímo nezadá. Nikdy totiž nevíte, jestli se obsah proměnné neskládá z něčeho, co uživatel zadal a jednoduše se to nedá ohlídat. Proto se ošetří vše a to **přímo na tom místě, kde proměnnou vypisujeme**. Až budete pokročilejší a budete znát objektovou architekturu, dokážete tento proces zutomatizovat, abyste funkci nemuseli ručně psát.

Ukažme si konečně upravený HTML kód formuláře i s direktivou nad ním:

```
<?php
        if ($hlaska)
                echo('<p>' . htmlspecialchars($hlaska) . '</p>');

        $jmeno = (isset($_POST['jmeno'])) ? $_POST['jmeno'] : '';
        $email = (isset($_POST['email'])) ? $_POST['email'] : '';
        $zprava = (isset($_POST['zprava'])) ? $_POST['zprava'] : '';
?>

<form method="POST">
        <table>
                <tr>
                        <td>Vaše jméno</td>
                        <td><input name="jmeno" type="text" value="<?=
htmlspecialchars($jmeno) ?>"/></td>
                </tr>
                <tr>
                        <td>Váš email</td>
                        <td><input name="email" type="email" value="<?=
htmlspecialchars($email) ?>"/></td>
                </tr>
                <tr>
                        <td>Aktuální rok</td>
                        <td><input name="rok" type="number" /></td>
                </tr>
        </table>
        <textarea name="zprava"><?= htmlspecialchars($zprava)
?></textarea>
        <br />

        <input type="submit" value="Odeslat" />
</form>
```

Formulář si vyzkoušejme. Ponechme nějaké pole prázdné a odešleme, ukáže se chybová hláška a zároveň zůstane i to, co uživatel vyplnil:



Přesměrování

Formulář má nyní dvě podstatné vady. Pokud zprávu úspěšně odešleme, stejně zůstane předvyplněný. A hlavně pokud po odeslání stiskneme F5, formulář se odešle znovu. Tímto neduhem trpí všechny formuláře, pokud jej zpracovává ten samý soubor, ve kterém je formulář vložený (což je většinou nutné).

Pokud je zpracování formuláře dokončeno, měli bychom provést přesměrování. Přesměrujeme na tu samou adresu, na které je formulář. Díky přesměrování se ovšem ztratí data v $_POST. Následné obnovení stránky tak již nic neodešle.

Přesměrování provedeme pomocí funkce header. Ta odešle tzv. hlavičku prohlížeči. Právě hlavička můžem obsahovat informaci o přesměrování a to pomocí slova Location.

Po uložení úspěšné hlášky tedy formulář přesměrujeme:

```
if ($uspech)
{
        $hlaska = 'Email byl úspěšně odeslán, brzy vám odpovíme.';
        header('Location: mailform.php');
        exit;
}
```

Funkcí exit() ukončíme běh skriptu, jelikož samotné přesměrování ho nezastaví, jen pošle prohlížeči návštěvníka informaci o tom, že se má přesunout na jinou lokaci.

Formulář již netrpí opětovným odesláním při obnovení stránky. Nezobrazuje ovšem ani hlášku o úspěchu. Mělo by vám být jasné proč, když přesměrujeme na jinou stránku, obsah proměnných na stránce stávající se ztratí a tím i ten v $hlaska. Řešení je několik, tím nejjednodušším je předat pomocí GET parametru stránce nějakou proměnnou. Podle

toho stránka pozná, že na ní bylo přesměrováno po úspěšném odeslání a zobrazí o tom zprávu. Kód změníme na následující podobu:

```
header('Location: mailform.php?uspech=ano');
exit;
```

Přesuneme se na začátek skriptu, kde hlášku nastavujeme na prázdný string. Podíváme se, zda nám nepřišel v adrese parametr úspěch. Pokud ano, nastavíme hlášku na text, který se má při úspěchu uživateli vypsat. Již víme, že parametry z URL adresy přicházejí na rozdíl od parametrů z formuláře v poli $_GET:

```
$hlaska = '';
if (isset($_GET['uspech']))
        $hlaska = 'Email byl úspěšně odeslán, brzy vám odpovíme.';
```

Můžete si vychutnat dobře fungující formulář.

POZOR! Přesměrovat můžete pouze v případě, že jste ještě nevypsali žádné HTML. Jakmile se totiž začne něco vypisovat, PHP prohlížeči odešle hlavičku, kde mu říká, že mu posílá HTML soubor. Hlavičku lze samozřejmě odeslat jen jednou, když se pokusíte veprostřed souboru přesměrovat, dostanete chybovou hlášku "Headers already sent" a k přesměrování nedojde. To by se stalo třeba v tomto případě:

```
<html>
        <body>
                <?php
                        // Tento kód je špatně
                        header('Location: index.php');
                        exit;
                ?>
        </body>
</html>
```

Dávejte si pozor i na tento případ, ve kterém chyba není na první pohled vidět:

```
-- Prázdná řádka --
<?php

        header('Location: index.php');

?>
<html>
        <body>
        </body>
</html>
```

Na začátku souboru je odřádkování. I to je znak, který odstartuje výstup a PHP ho odesílá prohlížeči spolu s hlavičkou. Podobný problém bývá s mezerou. Pokud ukládáte

UTF-8 soubory s tzv. BOM, může dělat právě tyto problémy. Pokud však používáte kvalitní IDE, tak se vám to nestane.

Pozn.: Někteří začátečníci přesměrovávají tak, že vyechují JavaScript, pomocí kterého změní adresu okna. Vypadá asi takto:

```
// Tento kód je špatně
echo('<script type="text/javascript">
        window.location = "index.php"
</script>');
```

Tento způsob je však nespolehlivý a někdy i nebezpečný. Nepoužívejte ho.

Formulář si můžete nastylovat, aby vypadal lépe. Hlášku dát do nějaké bubliny a podobně. To ale již není předmětem tohoto seriálu, stylování se probírá v jiné sekci Kompletní formulář je ke stažení níže. Příště se podíváme na skládání stránek pomocí PHP.

Príklad mailform2.php

```
<?php

        mb_internal_encoding("UTF-8");

    $hlaska = '';
    if (isset($_GET['uspech']))
        $hlaska = 'Email byl úspěšně odeslán, brzy vám odpovíme.';
    if ($_POST) // V poli _POST něco je, odeslal se formulář
    {
        if (isset($_POST['jmeno']) && $_POST['jmeno'] &&
            isset($_POST['email']) && $_POST['email'] &&
            isset($_POST['zprava']) && $_POST['zprava'] &&
            isset($_POST['rok']) && $_POST['rok'] == date('Y'))
        {
            $hlavicka = 'From:' . $_POST['email'];
            $hlavicka .= "\nMIME-Version: 1.0\n";
            $hlavicka .= "Content-Type: text/html; charset=\"utf-8\"\n";
            $adresa = 'vas@email.cz';
            $predmet = 'Nová zpráva z mailformu';
            $uspech = mb_send_mail($adresa, $predmet, $_POST['zprava'],
$hlavicka);
            if ($uspech)
            {
                $hlaska = 'Email byl úspěšně odeslán, brzy vám odpovíme.';
                header('Location: mailform.php?uspech=ano');
                exit;
            }
            else
```

```
                $hlaska = 'Email se nepodařilo odeslat. Zkontrolujte
adresu.';
        }
        else
            $hlaska = 'Formulář není správně vyplněný!';
    }

?>

<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Kontaktní formulář</title>
    </head>
    <body>
        <p>Můžete mě kontaktovat pomocí formuláře níže.</p>

        <?php
            if ($hlaska)
                echo('<p>' . htmlspecialchars($hlaska) . '</p>');

            $jmeno = (isset($_POST['jmeno'])) ? $_POST['jmeno'] : '';
            $email = (isset($_POST['email'])) ? $_POST['email'] : '';
            $zprava = (isset($_POST['zprava'])) ? $_POST['zprava'] : '';
        ?>

        <form method="POST">
            <table>
                <tr>
                    <td>Vaše jméno</td>
                    <td><input name="jmeno" type="text" value="<?=
htmlspecialchars($jmeno) ?>"/></td>
                </tr>
                <tr>
                    <td>Váš email</td>
                    <td><input name="email" type="email" value="<?=
htmlspecialchars($email) ?>"/></td>
                </tr>
                <tr>
                    <td>Aktuální rok</td>
                    <td><input name="rok" type="number" /></td>
                </tr>
            </table>
            <textarea name="zprava"><?= htmlspecialchars($zprava)
?></textarea>
            <br />

            <input type="submit" value="Odeslat" />
```

```
        </form>

    </body>
```

</html>

10. díl - Skládání stránek v PHP

Dynamické skládání stránek

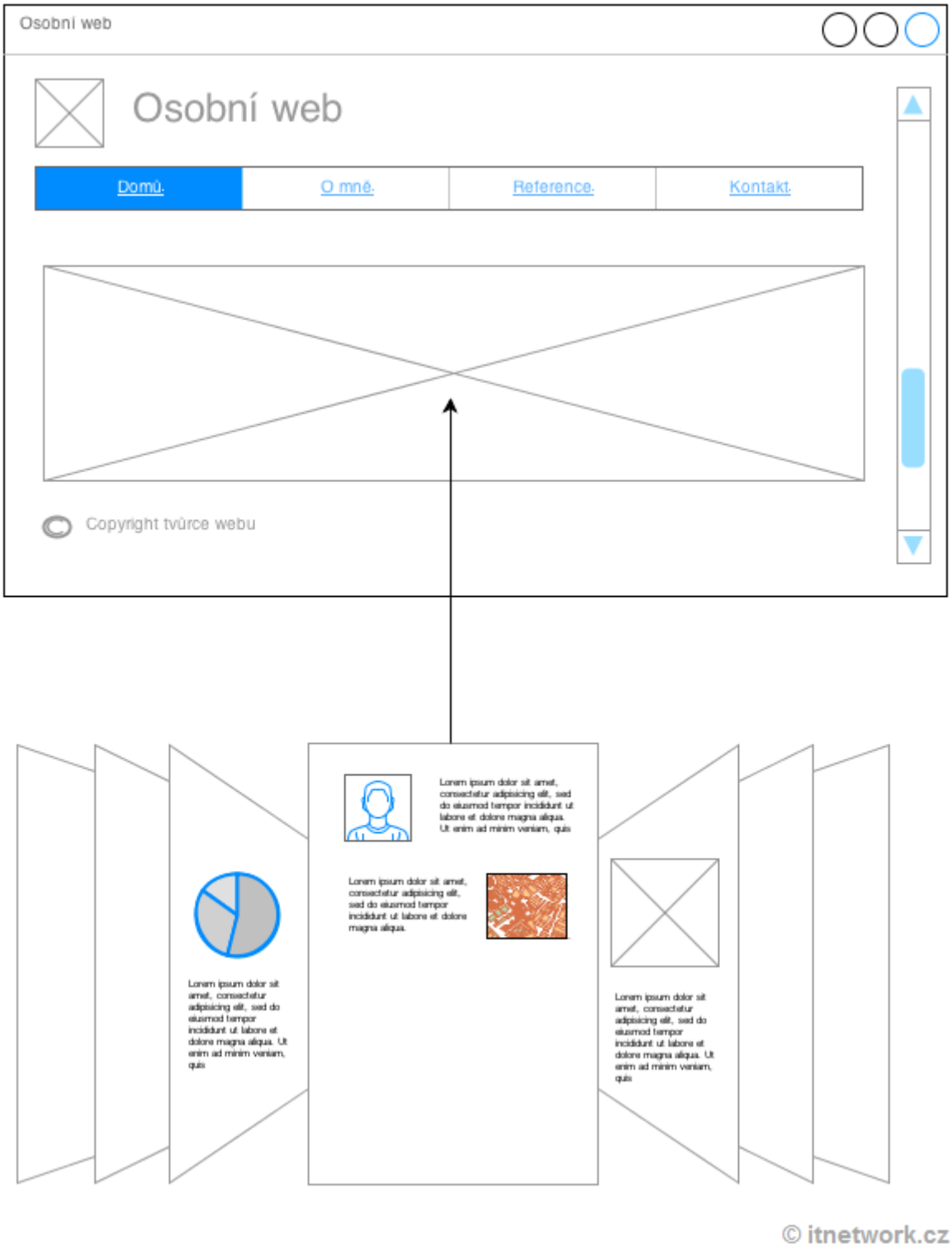
Většina dnešních webových stránek se skládá vlastně ze 2 částí. Tou první je tzv. layout, neboli rozložení webu. To je na každé podstránce webu stejné a obvykle obsahuje logo webu, navigační menu a patičku. Druhá část stránky je potom samotný článek, který je do layoutu zobrazen.

© itnetwork.cz

U statických webových stránek jsme do každé podstránky museli layout ručně zkopírovat (v minulosti se daly používat rámce, ale ty byly z HTML kvůli četným problémům odebrány). Mít v každé podstránce celý layout je samozřejmě nepřehledné, pracně se tam vkládá a hlavně je problém v layoutu potom něco změnit, změnu musíme udělat ve všech podstránkách.

Jelikož nyní známe PHP, není nejmenší problém v tom, aby obsah článku do layoutu vložilo za nás. Na webu budeme mít v jednotlivých souborech pouze podstránky (bez layoutu) a layout bude přítomný v souboru index.php. Zde do layoutu vložíme ten článek, který uživatel vyžaduje. Uživatel si o článek řekne metodou GET, vloží tedy jméno stránky do URL adresy.

Můžeme si to představit takto:



© itnetwork.cz

Způsoby skládání stránek

Stránky můžeme skládat vlastně dvěma způsoby. Buď jak jsme si uvedli výše, že do layoutu vkládáme podstránku, nebo můžeme do podstránky vložit nahoru hlavičku a dolů patičku. My si zde ukážeme vkládání podstránky do layoutu, jelikož tento princip se dále používá i pro vkládání článků z databáze.

Příprava souborů

Založme si nový projekt a připravme si potřebné soubory. Budeme potřebovat index.php, ve kterém se bude nacházet náš layout. Já jsem níže uvedený layout převzal ze zdejšího HTML seriálu, můžete si ho odtamtud vypůjčit a stáhnout i potřebný styl a obrázky. Nebo si prostě napište svůj, bohatě stačí, když tam bude nadpis 🙂

```
<!DOCTYPE html>
<html lang="cs-cz">

	<head>
		<meta charset="utf-8" />
		<link rel="stylesheet" href="styl.css" type="text/css" />
		<title>HoBiho portfolio</title>
	</head>

	<body>
		<header>
			<div id="logo"><h1>HoBi</h1></div>
			<nav>
				<ul>
					<li><a
href="index.php?stranka=domu">Domů</a></li>
					<li><a
href="index.php?stranka=omne">O mně</a></li>
					<li><a
href="index.php?stranka=dovednosti">Dovednosti</a></li>
					<li><a
href="index.php?stranka=reference">Reference</a></li>
					<li><a
href="index.php?stranka=kontakt">Kontakt</a></li>
				</ul>
			</nav>
		</header>

		<article>
			<div id="centrovac">
				<header>
					<h1>O mně</h1>
```

```
                        </header>

                        <section>
                                <?php

                                ?>
                        </section>
                        <div class="cistic"></div>
                </div>
        </article>


        <footer>
                Vytvořil &copy;HoBi 2013 pro <a
href="http://devbook.cz">DEVBOOK.CZ</a>
        </footer>
    </body>
</html>
```

V layoutu máme nějakou HTML hlavičku, dále hlavičku webové stránky, ve které je logo a navigace. Všimněte si, že odkazy na podstránky směřují na adresu:

```
index.php?stranka=domu
```

Všechny odkazy tedy vedou na layout, kterému předávají v parametru název stránky, která se do něj má vložit.**Tyto HTML stránky si vložíme do podsložky podstranky se stejným názvem a příponou .php**. Podstránka omne tedy bude uložena v:

```
podstranky/omne.php
```

Podstránky by teoreticky mohly mít i příponu HTML, ale často v některé budeme chtít nějaký PHP skript, např. pro kontaktní formulář.

Všimněte si také prázdné PHP sekvence v elementu section. Přesně tam budeme podstránku vkládat.

Vytvořte si složku podstranky a několik takových podstránek s příponou .php.

Vložení souboru

PHP má víceméně dva způsoby, jak do nějakého skriptu vložit obsah jiného souboru.

Vložení textu

Pokud chceme obsah cizího souboru vložit jako text, slouží k tomu PHP funkce file_get_contents(). Funkce bere v parametru cestu k souboru a vrací text, který soubor obsahuje.

Obsah podstránky bychom pomocí funkce vložili kódem níže. **Kód není ještě bezpečný, což dořešíme na konci článku**.

```php
// skript níže není bezpečný
$obsah = file_get_contents('podstranky/' . $_GET['stranka'] . '.html');
echo $obsah;
```

Všimněte si, že jsem dal podstránkám příponu .html. Pokud bychom měli v podstránce totiž PHP skript a vložili ho touto funkci do layoutu, vypsal by se zdrojový kód skriptu místo toho, aby se provedl. To může být dost nebezpečné, jelikož PHP skripty běžně obsahují přístupové údaje k databázi a další citlivá data. Pokud však chceme vložit nějaké HTML nebo prostý text z nějakého souboru, je funkce ideální.

Dalším bezpečnostním problémem je, že uživatel si může do parametru napsat co chce a vypisovat si tak i obsah stránek, které ukazovat nechceme.

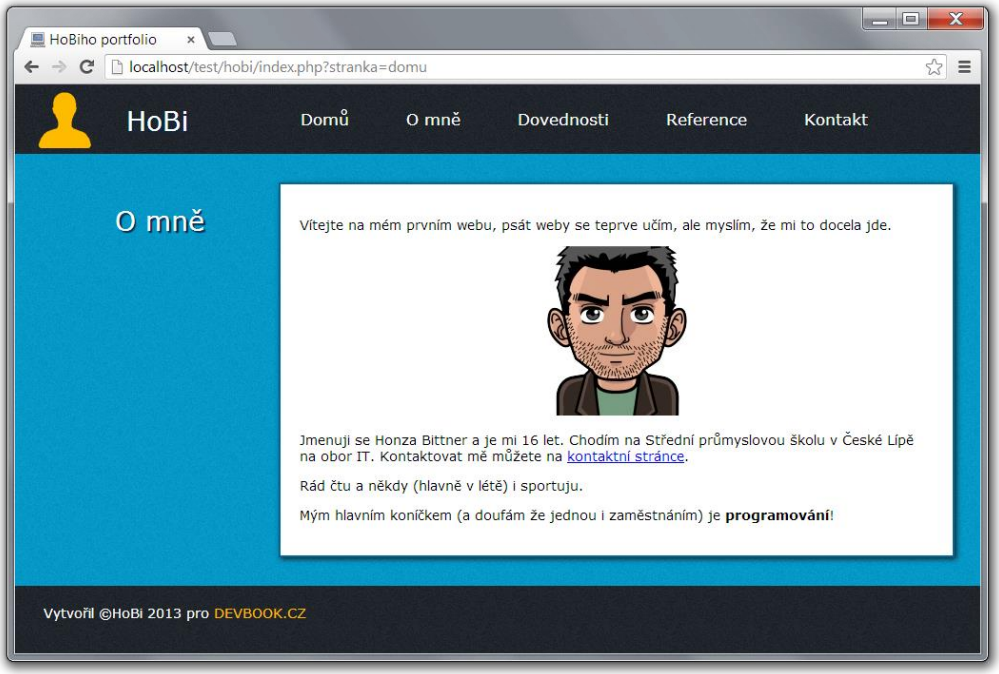
Vložení skriptu

Pokud chceme, aby se obsah cizího souboru vykonal jako PHP skript, použijeme k tomu funkci require(). Funkce bere jako parametr opět cestu k souboru a obsah hned vypíše. Pokud je v souboru nějaká PHP sekvence, tak ji spustí.

Do PHP sekvence v index.php vložíme následující kód:

```php
// skript níže není bezpečný
require('podstranky/' . $_GET['stranka'] . '.php');
```

A vyzkoušíme zadat adresu nějaké podstránky:



Můžete si zkusit proklikat menu, podstránka se vždy vloží do layoutu a celý web se vypíše v prohlížeči. Velmi jsme si zjednodušili práci a otevřeli cestu do budoucna, kdy texty budeme vkládat z databáze.

Pozn. Kromě require nalezneme v PHP i funkci include. Ta funguje úplně stejně, jen při neúspěšném vložení nezastaví běh aplikace, ale pouze vyvolá warning..

Bezpečnost

POZOR! Obě výše zmíněné funkce jsou potenciálně velmi nebezpečné a pokud je špatně použijete, vznikne vám v aplikaci bezpečnostní díra.

Podstata problému je v tom, že zobrazujeme obsah nějakého souboru, jehož název zadává uživatel. Ten tak může načíst např. soubor .htpasswd, ve kterém jsou uložena hesla a to touto adresou:

```
http://vasestranka.cz/index.php?stranka=../.htpasswd
```

Sekvence ../ přesune o složku výše. Dostaneme se tedy z podstránek do kořenové složky s webem. Následně útočník může zobrazit obsah úplně čehokoli.

Zabezpečení

Řešením této bezpečnostní trhliny je samozřejmě ošetřit vstup uživatele tak, aby mohl obsahovat jen znaky a-z a maximálně čísla. To uděláme pomocí tzv. regulárního výrazu. Teorie okolo těchto výrazů je poměrně složitá, ale jednoduše řečeno se jedná o takový "minijazyk" (správně metajazyk), který slouží zejména pro kontrolu obsahu textových

řetězců. V dalších sekcích webu se jim ještě budeme věnovat, nyní nám musí stačit, že k ověření řetězce pomocí regulárního výrazu slouží PHP funkce preg_match(), která vrátí 1 pokud text odpovídá.

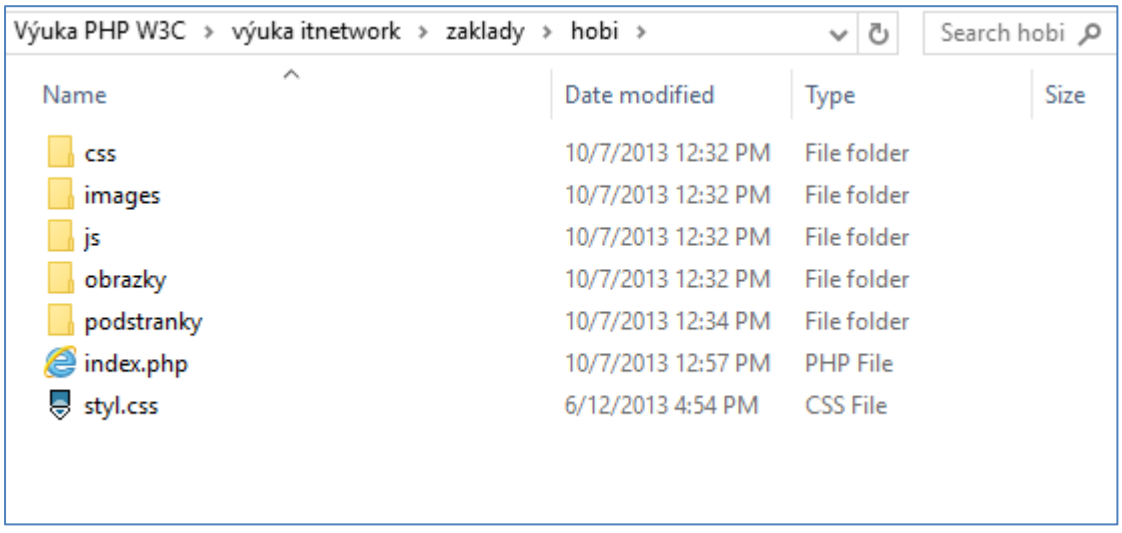
Rovnou i změníme require na include a pokud se vložení nepovede, zobrazíme chybovou hlášku. U require se takto ptát nemůžeme, při neúspěchu vždy zastaví aplikaci. Také přidáme podmínku, že pokud parametr není zadaný, zobrazí se podstránka domů.

```
if (isset($_GET['stranka']))
        $stranka = $_GET['stranka'];
else
        $stranka = 'domu';
if (preg_match('/^[a-z0-9]+$/', $stranka))
{
        $vlozeno = include('podstranky/' . $stranka . '.php');
        if (!$vlozeno)
                echo('Podstránka nenalezena');
}
else
        echo('Neplatný parametr.');
```

Web je nyní bezpečný a vás již nebrzdí ruční kopírování layoutu. Můžete si zkusit vložit jako podstránku náš kontaktní formulář.

Nad otázkou bezpečnosti bychom se u webových aplikací měli zamýšlet vždy, když uživatel někde něco zadává. Vždy se musíme zeptat, co se s touto hodnotou děje a zda nemůže nějaký vstup způsobit bezpečnostní chybu.

Hotový web je v příloze ke stažení, příště se podíváme na cykly. Brzy se také dostaneme k databázím.



11. díl - Cykly for a while v PHP

Cyklus

Ako už slovo cyklus napovie, niečo sa bude opakovať. Keď chceme v programe niečo urobiť 100x, určite nebudeme písať pod seba 100x ten istý kód, ale vložíme ho do cyklu. Cyklov máme niekoľko druhov,FOR a WHILE.

FOR cyklus

Tento cyklus má stanovený pevný počet opakovaní a hlavne obsahuje tzv. Riadiacu premennú (celočíselnú), v ktorej sa postupne počas behu cyklu menia hodnoty. Syntax (zápis) cykle for je nasledovná:

```
for (premenná; podmienka; prikaz)
```

- **premenna** je riadiaca premenná cyklu, ktoréj nastavíme počiatočnú hodnotu (najčastejšie 0, už vieme, že v programovaní všetko začína od nuly, nie od jednotky). Napr. teda $i = 0. Býva zvykom používať názov aj, ako index.
- **Podmienka** je podmienka vykonanie ďalšieho kroku cyklu. Akonáhle nebude platiť, cyklus sa ukončí. Podmienka môže byť napr ($ i <10).
- **prikaz** nám hovorí, čo sa má v každom kroku s riadiacou premennou stáť. Teda či sa má zvýšiť alebo znížiť. K tomu využijeme špeciálnych príkazov ++ a --, tie samozrejme môžete používať aj bežne mimo cyklus, slúži k zvýšeniu alebo zníženiu premennej o 1.

Poďme si urobiť jednoduchý príklad, väčšina z nás určite pozná Sheldona z The Big Bang Theory. Pre tých čo nie, budeme simulovať situáciu, kedy klope na dvere svojej susedky. Vždy 3x zaklope a potom zavolá: "Penny!". Náš kód by bez cyklov vyzeral takto:

```
echo('Knock<br />');
echo('Knock<br />');
echo('Knock<br />');
echo('Penny!');
```

knock.php

```
<?php
        echo('Knock<br />');
        echo('Knock<br />');
        echo('Knock<br />');
        echo('Penny!');
```

```
        ?>
```

Php server

Knock
Knock
Knock
Penny!

My ale už nič nemusíme otrocky opisovať,použijeme cyklus

Knock3.php

```
<?php
        for ($i=0; $i < 3; $i++)
                {
         echo('Knock<br />');
                }
echo('Penny!');
?>
```

Výsledok:

Knock
Knock
Knock
Penny!

Cyklus prebehne 3x, spočiatku je v premennej $i nula, cyklus vypíše "Knock" a zvýši premennú $i o jedna. Potom beží rovnako s jednotkou a dvojkou. Akonáhle je v $i trojka, už nesúhlasí podmienka $ i <3 a cyklus končí.

O vynechávaní zložených zátvoriek platí to isté, čo u podmienok. V tomto prípade tam nemusí byť, pretože cyklus spúšťa iba jediný príkaz.

Teraz môžeme miesto trojky napísať do deklarácie cyklu desiatku. Príkaz sa spustí 10x bez toho aby sme písali niečo naviac. Určite vidíte, že cykly sú mocným nástrojom.

Knock10.php

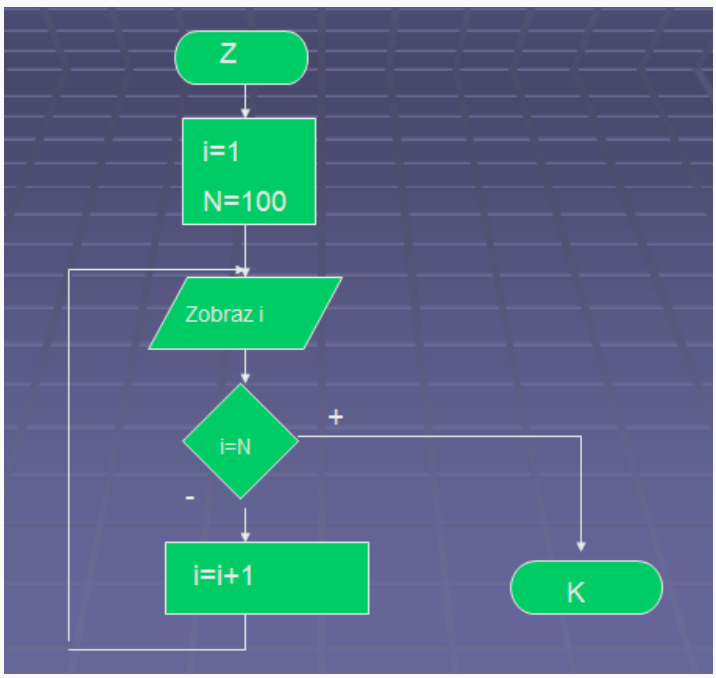
```php
<?php
        for ($i=0; $i < 10; $i++)
                {
         echo('Knock<br />');
                }
echo('Penny!');
?>
```

Server php

Knock
Knock
Knock
Knock
Knock
Knock
Knock
Knock
Knock
Knock
Penny!

Skúsme si teraz využiť toho, že sa nám premenná inkrementuje. Vypíšme si čísla od jednej do sto:

Grafický zápis



for100.php

```
<?php
for ($i = 1; $i <= 100; $i++)
        echo($i . ' ');
?>
```

Server php

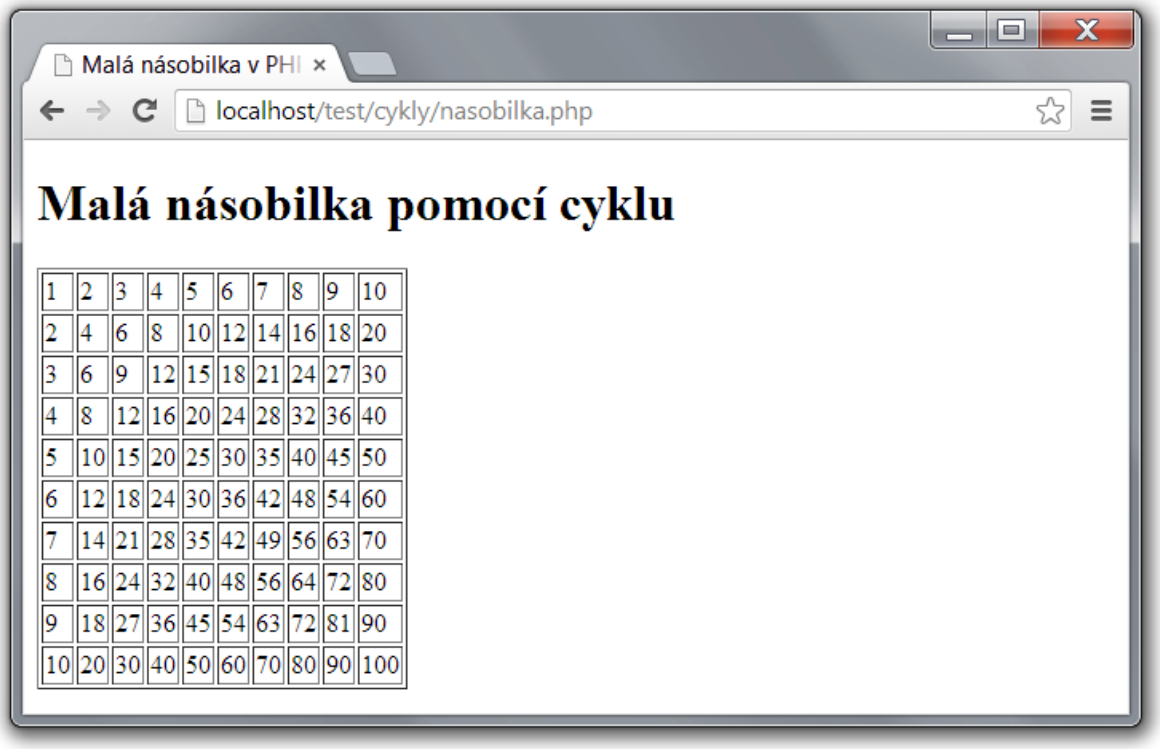
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Vidíme, že riadiaci premenná má naozaj v každej iterácii (priebehu) inú hodnotu. Všimnite si, že v cykle tentoraz nezačíname na nule, ale môžeme nastaviť počiatočnú hodnotu na 1 a koncovú na 100. V programovaní je však zvykom začínať od nuly, pretože sa od nuly indexujú poľa.

```
Teraz si vypíšeme malú násobilku (násobky čísel 1 až 10, vždy do desať).
Stačí nám urobiť cyklus od 1 do 10 a premennú vždy násobiť daným číslom.
Mohlo by to vyzerať asi takto:

echo('<h1>Malá násobilka pomocí cyklu</h1>');
echo('<table border="1"><tr>');
for ($i = 1; $i <= 10; $i++)
        echo('<td>' . $i . '</td>');
echo('</tr><tr>');
for ($i = 1; $i <= 10; $i++)
        echo('<td>' . ($i * 2) . '</td>');
echo('</tr><tr>');
for ($i = 1; $i <= 10; $i++)
        echo('<td>' . ($i * 3) . '</td>');
echo('</tr><tr>');
for ($i = 1; $i <= 10; $i++)
        echo('<td>' . ($i * 4) . '</td>');
echo('</tr><tr>');
for ($i = 1; $i <= 10; $i++)
        echo('<td>' . ($i * 5) . '</td>');
echo('</tr><tr>');
for ($i = 1; $i <= 10; $i++)
        echo('<td>' . ($i * 6) . '</td>');
echo('</tr><tr>');
for ($i = 1; $i <= 10; $i++)
        echo('<td>' . ($i * 7) . '</td>');
echo('</tr><tr>');
for ($i = 1; $i <= 10; $i++)
```

```
        echo('<td>' . ($i * 8) . '</td>');
echo('</tr><tr>');
for ($i = 1; $i <= 10; $i++)
        echo('<td>' . ($i * 9) . '</td>');
echo('</tr><tr>');
for ($i = 1; $i <= 10; $i++)
        echo('<td>' . ($i * 10) . '</td>');
echo('</tr></table>');
```

Výstup:



1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

Program funguje pekne, ale stále sme toho dosť napísali. Ak vás napadlo, že v podstate robíme 10x to isté a len zvyšujeme číslo, ktorým násobíme, máte pravdu. Nič nám nebráni vložiť 2 cykly do seba:

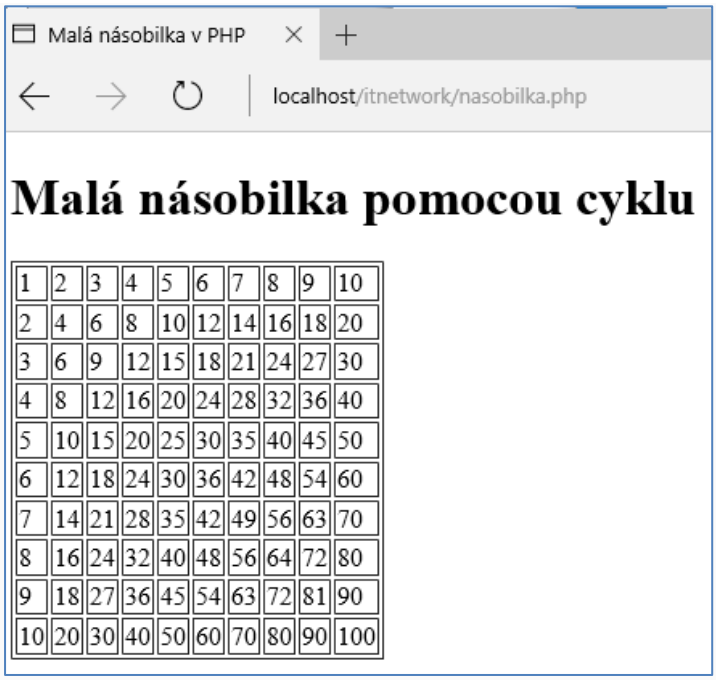
Kostra príkladu

```
echo('<h1>Malá násobilka pomocí cyklu</h1>');
echo('<table border="1">');
for ($j = 1; $j <= 10; $j++)
{
        echo('<tr>');
        for ($i = 1; $i <= 10; $i++)
                echo('<td>' . ($i * $j) . '</td>');
        echo('</tr>');
}
echo('</table>');
```

Priklad nasobilka.php

```
<!DOCTYPE html>
<html lang="sk">
    <head>
        <meta charset="UTF-8">
        <title>Malá násobilka v PHP</title>
    </head>
    <body>
        <?php


            echo('<h1>Malá násobilka pomocou cyklu</h1>');
            echo('<table border="1">');
            for ($j = 1; $j <= 10; $j++)
            {
                echo('<tr>');
                for ($i = 1; $i <= 10; $i++)
                    echo('<td>' . ($i * $j) . '</td>');
                echo('</tr>');
            }
            echo('</table>');

        ?>
    </body>
</html>
```

Výstup:



Pomerne zásadný rozdiel, že? Pochopiteľne nemôžeme použiť u oboch cyklov $i, pretože sú vložené do seba.

Premenná $j nadobúda vo vonkajšom cykle hodnoty 1 až 10. V každej iterácii (rozumejte priebehu) cyklu je potom spustený ďalší cyklus s premennou $i.

Ten je nám už známy, vypíše násobky, v tomto prípade násobíme premennú $ j. Po každom behu vnútorného cyklu je tiež potrebné otvoriť a ukončiť nový riadok.

Urobme si ešte jeden program, na ktorom si ukážeme prácu s vonkajšou premennou.
Aplikácia bude vedieť spočítať ľubovoľnú mocninu ľubovoľného čísla:

Kostra príkladu:

```
$a = 2; // základ mocniny
$n = 3; // exponent

$vysledek = $a;
for ($i = 0; $i < ($n - 1); $i++)
        $vysledek = $vysledek * $a;

echo("Výsledek: $vysledek");
```

priklad mocniny.php

```
<!DOCTYPE html>
<html lang="sk">
    <head>
        <meta charset="UTF-8">
        <title>Mocniny</title>
    </head>
    <body>
        <?php
            $a = 2; // základ mocniny
            $n = 3; // exponent

            $vysledok = $a;
            for ($i = 0; $i < ($n - 1); $i++)
                $vysledok = $vysledok * $a;

            echo("Výsledok: $vysledok");

        ?>
    </body>
</html>
```

Výstup:

Asi všetci tušíme, ako funguje mocnina. Pre istotu pripomeniem, že napríklad 2^3 = 2 * 2 * 2. Teda a^n spočítame tak, že n-1 krát vynásobíme číslo $a číslom $a. Výsledok si samozrejme musíme ukladať do premennej. Spočiatku bude mať hodnotu $a a postupne sa bude v cykle prenásobovat. Ak ste to nestihli, máme tu samozrejme článok s algoritmom výpočtu ľubovoľnej mocniny. Vidíme, že naša premenná $vysledek je v tele cyklu normálne prístupná. Ak si však nejakú premennú založíme v tele cykle, po skončení cyklu zanikne a už nebude prístupná.

```
Aplikácia bude vedieť spočítať ľubovoľnú mocninu ľubovoľného čísla:
```

Príklad výpočtu ľubovoľného základu a ľubovoľného eponentu.

Súbor mocniny2.html

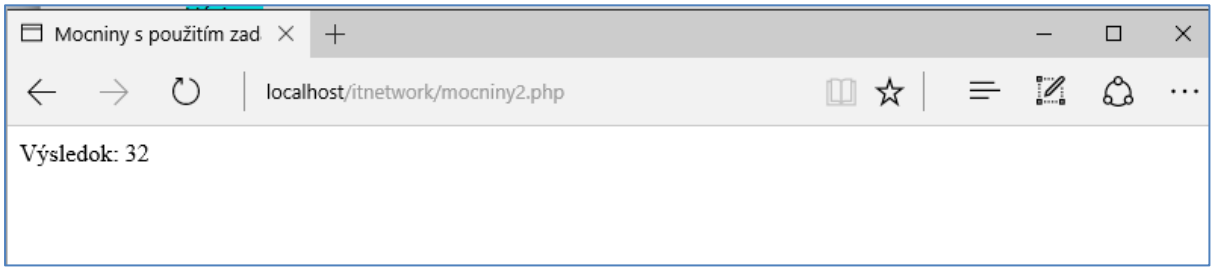
```
<!DOCTYPE html>
<html lang="sk">
    <head>
        <meta charset="UTF-8">
        <title>Zadavacia sekvencia pre php script mocniny2.php</title>
    </head>
    <body>
        <p>Pre výpočet mocniny zadajte základ mocniny a exponent.</p>

        <form method="POST" action="mocniny2.php">
         Zaklad:     <input name="zaklad" type="number" /><br />
         Exponent:   <input name="exponent" type="number" /><br />
         <br />
            <input type="submit" value="Vypočítaj mocninu!!!" />
        </form>

    </body>
</html>
```

Súbor mocniny2.php

```
<!DOCTYPE html>
<html lang="sk">
    <head>
        <meta charset="UTF-8">
        <title>Mocniny s použitím zadavacieho poľa</title>
    </head>
```

```
    <body>
        <?php


        $a = $_POST['zaklad'];//základ mocniny
        $n = $_POST['exponent'];//exponent
               $vysledok = $a;
            for ($i = 0; $i < ($n - 1); $i++)
                $vysledok = $vysledok * $a;

            echo("Výsledok: $vysledok");

        ?>
    </body>
</html>
```

Výstup:



While cyklus

While cyklus funguje inak, jednoducho opakuje príkazy v bloku kým platí podmienka. Syntax cyklu je nasledujúci:

```
while (podmienka)
{
príkazy
```

Ak vás napadá, že možno cez while cyklus urobiť aj FOR cyklus, máte pravdu, FOR je vlastne špeciálny prípad while cyklu.

While sa ale používa na trochu iné veci, často máme v jeho podmienke napr. Funkciu vracajúci logickú hodnotu true / false.

Pôvodný príklad z for cyklu by sme urobili nasledovne pomocou while:}

While1.php

```php
<?php
$i = 1;
while ($i <= 10)
{
        echo($i . ' ');
        $i++;
}
?>
```

Server php

1 2 3 4 5 6 7 8 9 10

To ale nie je ideálne použitie while cyklu. While sa používa najmä v prípadoch, keď čítame zo súborov a nevieme, kedy narazíme na posledný riadok

Uloha :Vytvorte script na výpočet faktorialu zadaného čísla

Poznánka:5! sa počíta :5*4*3*2*1,t.j.120

Grafický zápis scriptov,pozor na iné značenie premenných

faktorial.html....tento treba zadať v localhoste,**nie php** subor

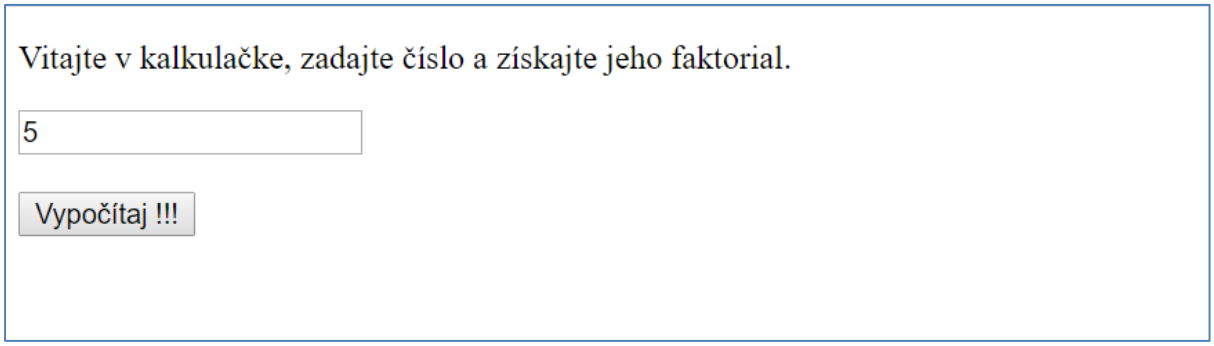
```
<!DOCTYPE html>
<html lang="sk">
    <head>
        <meta charset="UTF-8">
        <title>Faktorial čísla</title>
    </head>
    <body>
        <p>Vitajte v kalkulačke, zadajte číslo a získajte jeho
faktorial.</p>

        <form method="POST" action="faktorial.php">
            <input name="cislo" type="number" /><br /><br>
            <input type="submit" value="Vypočítaj !!!" />
        </form>

    </body>
</html>
```

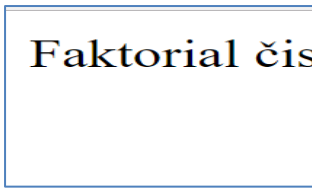
faktorial.php

```
<!DOCTYPE html>
<html lang="sk">
    <head>
        <meta charset="UTF-8">
        <title>Faktorial čísla</title>
    </head>
    <body>
        <?php
        $suma = 1;
        $pocitadlo = 1;
        $fakt = $_POST['cislo'] ;
```

```
            while($pocitadlo<=$fakt)
                {
                $suma = $suma * $pocitadlo;
                $pocitadlo ++;
                }

        echo("Faktorial čisla:$fakt je $suma");
        ?>
    </body>
</html>
```

Server php :





Cyklus do.....while

Dowhile.php

```
<!DOCTYPE html>
<html lang="sk">
    <head>
        <meta charset="UTF-8">
        <title>Gyklus do while</title>
    </head>
    <body>
    <?php
          $c = 1;
            do {
              echo("$c<br>");
```

```
                $c++;
                }
                while ($c <101);
                 ?>
  </body>
</html>
```

Php server

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
```

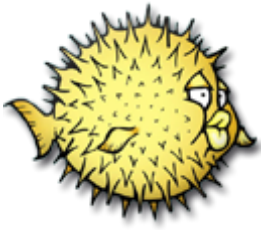
Cvičenia k cyklom PHP

Nasledujúce 3 cvičenia vám pomôžu precvičiť znalosti programovania v PHP z minulej lekcie. Vo vlastnom záujme sa je pokúste vyriešiť sami. Pod článkom máte na kontrolu riešenie k stiahnutiu. Ale pozor, akonáhle sa na neho pozriete bez vyriešenia príkladov, stráca pre vás cvičenie zmysel a nič sa nenaučíte

Jednoduchý príklad

Vytvorte skript, ktorý sa používateľa pomocou formulára opýta, koľko si dá rýb na večeru. Potom mu ich zobrazí ako obrázky.

Použite nasledujúci obrázok ryby:



Priklad ryby.html

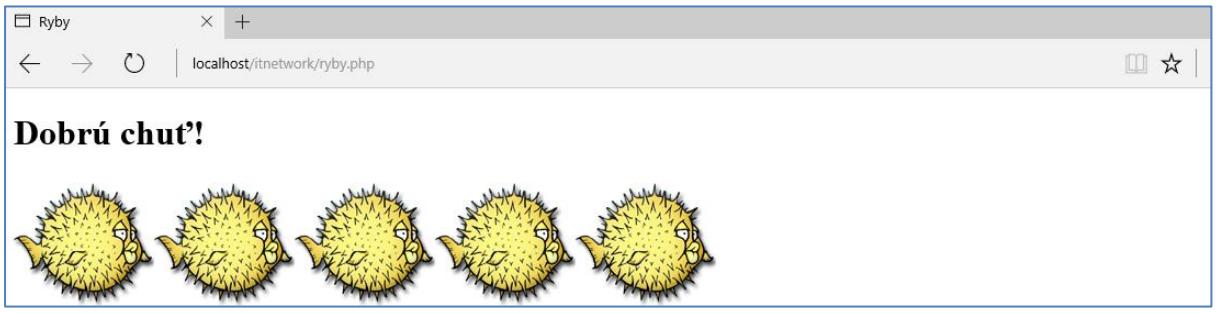
```
<!DOCTYPE html>
<html lang="sk">
<head>
    <meta charset="utf-8" />
    <title>Ryby</title>
</head>

<body>

<h1>Koľko rýb si dáš k večery?</h1>

<form action="ryby.php" method="post">
    <input type="number" name="pocet" /><br />
    <input type="submit" value="Nakŕmte ma !" />
</form>

</body>

</html>
```

Priklad ryby.php

```
<!DOCTYPE html>
<html lang="sk">
<head>
        <meta charset="utf-8" />
        <title>Ryby</title>
</head>

<body>

        <h1>Dobrú chuť!</h1>

        <?php
```

```
                $pocet = $_POST['pocet'];
                for ($i = 0; $i < $pocet; $i++)
                {
                        echo('<img src="ryba.png" alt="Ryba" />');
                }

        ?>

</body>

</html>
```

Výstup:



Stredne pokročilý príklad

Zadanie tohto skriptu je odvodené z anglickej riekanky, ktorá začína takto:

10 zelených fliaš stojí na stole a jedna fľaša spadne

Program ďalej pokračuje takto:

9 zelených fliaš stojí na stole a jedna fľaša spadne

Až skončí posledný vetou:

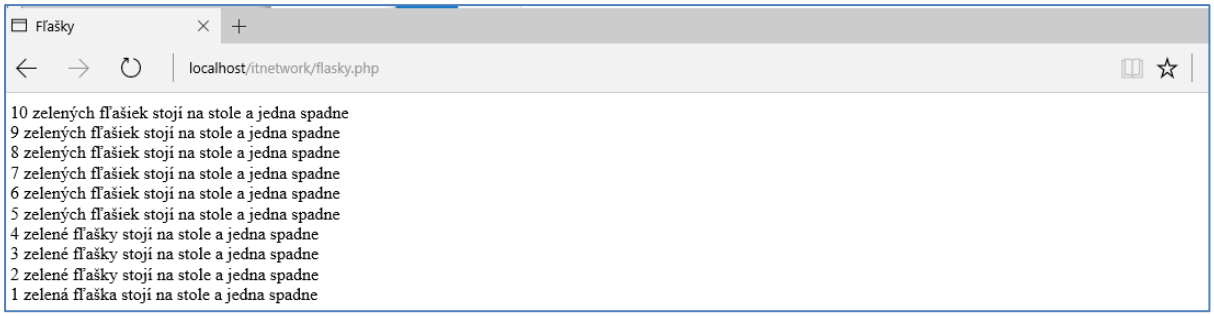
1 zelená fľaša stojí na stole a jedna fľaša spadne

Vytvorte skript, ktorý vykoná takýto výstup pre 10 fliaš. Všimnite si, že skript vie skloňovať slová zelená a fľaša.

Priklad flasky.php

```
<!DOCTYPE html>
<html lang="sk">
<head>
        <meta charset="utf-8" />
        <title>Fľašky</title>
</head>

<body>

        <?php


                for ($i = 10; $i > 0; $i--)
                {
                        $tvar = "zelených fľašiek";
                        if (($i > 1) && ($i < 5))
                        {
                                $tvar = "zelené fľašky";
                        }
                        if ($i == 1)
                        {
                                $tvar = "zelená fľaška";
                        }
                        echo("$i $tvar stojí na stole a jedna spadne<br
/>");
                }
        ?>

</body>

</html>
```

Výstup



Pokročilý príklad

Napíšte skript pre vykreslenie šachovnice. Môžete využiť nasledujúce nápovedí:
1. K zobrazeniu šachovnice použite HTML tabuľku a štýlovanie pozadia buniek na čiernu / bielu farbu.
2. Určite budete chcieť využiť cyklov. Uvedomte si, že nie je problém umiestniť cyklus do cyklu a to napríklad tak, aby jeden echoval riadky a druhý bunky do týchto riadkov.
3. Či je číslo párne zistíte pomocou nulového zvyšku po delení, ktorý získate operátorom "%". Podmienka či je $ x deliteľné 2-mi je teda nasledujúca:

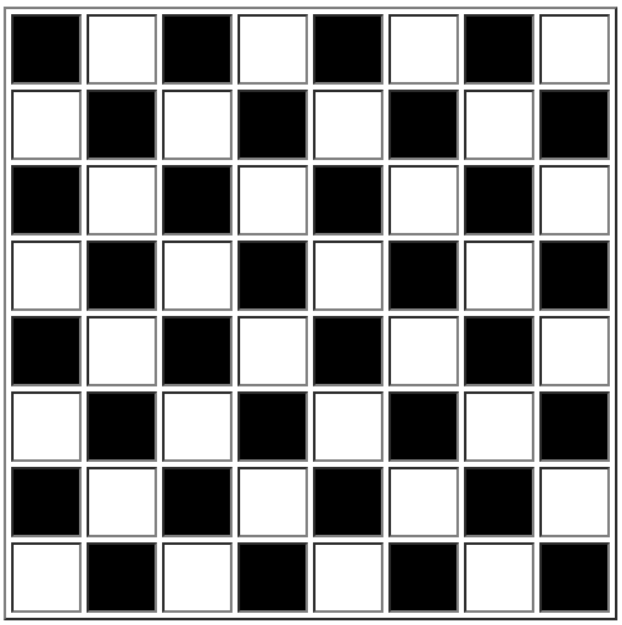
```
if ($x % 2 == 0)
```

Príklad sachovnica.php

```
<!DOCTYPE html>
<html lang="sk">
<head>
        <meta charset="utf-8" />
        <title>Šachovnica</title>
</head>

<body>

        <?php


                echo('<table border="1">');
                for ($i = 0; $i < 8; $i++)
                {
                        echo('<tr>');

                        for ($j = 0; $j < 8; $j++)
                        {
                                if(($i + $j) % 2 == 0) // Je párne
                                        $styl = 'style="background: black;';
```

```
                                else
                                        $styl = '';
echo('<td ' . $styl . ' width: 24px; height: 24px; color: white;" ></td>');
                            }
                            echo('</tr>');
                    }
                    echo('</table>');
            ?>

</body>

</html>
```

Výstup:



12. díl - Práce s polem pomocí cyklů v PHP

minulém dílu seriálu tutoriálů se základy PHP jsme se naučili cykly for a while. V ndešním dílu cykly dokončíme a ukážeme si, jak je lze využít při práci s polem.

Naplnění pole cyklem

Právě cykly se často používají pro automatizovanou práci s poli. Položek je v poli mnoho a pracovat s nimi po jedné by určitě nebyl nejlepší nápad. Začněme naplněním pole čísly od 1 do 100.

Chceme něco 100x opakovat, počet opakování tedy víme a proto zvolíme for cyklus. Řídící proměnná for cyklu ($i) bude nabývat hodnot 0 až 99, jelikož víme, že pole o 100 položkách má indexy 0 - 99. Protože chceme hodnoty v položkách od 1 do 99, musíme k indexu přičíst jedničku:

```
$cisla = array();
for ($i = 0; $i < 100; $i++)
        $cisla[$i] = $i + 1;
```

Výpis pole cyklem

Nyní máme v poli uložených 100 čísel. Podobné generování pole se obvykle v praxi moc nepoužívá, jelikož hodnoty v něm získáme např. z databáze. S něčím ale pracovat musíme  Co už se používá velmi často je výpis hodnot z pole, např. do tabulky.

Cyklus for

Určitě byste dali dohromady následující kód, který by patřil za kód výše uvedený:

```
echo('<table border="1"><tr>');
for ($i = 0; $i < 100; $i++)
        echo('<td>' . htmlspecialchars($cisla[$i]) . '</td>');
echo('</tr></table>');
```

Kód vypíše pomocí cyklu for obsah pole do tabulky. V poli by v praxi nebyla jen vygenerovaná čísla, ale jednalo by se např. o komentáře z databáze. K těm se brzy dostaneme.

Cyklus foreach

K výpisu pole se však mnohem více hodí zjednodušená verze cyklu for, kterou je foreach. Zapisujeme ho takto:

```
foreach($kolekce as $prvek)
```

Foreach proiteruje všechny položky v poli a v proměnné cyklu vždy vrací aktuální položku. To je rozdíl oproti cyklu for, který v proměnné cyklu vrací index aktuální položky.

Výpis obsahu našeho pole do tabulky bychom pomocí foreach napsali takto:

```php
echo('<table border="1"><tr>');
foreach($cisla as $cislo)
        echo('<td>' . htmlspecialchars($cislo) . '</td>');
echo('</tr></table>');
```

Výstup programu bude identický. Foreach slouží zejména pro čtení nebo pro práci s objekty (ty zatím ještě neznáme), kde je přehlednější, než cyklus for.

Proměnná cyklu obsahuje v příkladu výše kopii prvku na dané pozici pole. Pokud bychom se pokusili v cyklu proměnnou $cislo modifikovat, nemělo by to na pole žádný vliv, jelikož bychom změnili pouze kopii daného prvku a ne prvek samotný. Pokud bychom chtěli např. všechny prvky v poli vynásobit dvěma, musíme použít for:

```php
for ($i = 0; $i < 100; $i++)
        $cisla[$i] = $cisla[$i] * 2;
```

Často ale chceme např. jen dvojnásobky vypsat, nikoli měnit původní pole. K tomu by nám foreach stačil.

PHP funkce pro práci s polem

PHP pro práci s poli nabízí spoustu hotových funkcí. Nebudeme je popisovat nijak podrobně, pouze si uvedeme seznam těch nejdůležitějších. Každá funkce odkazuje na český PHP manuál, kde je popsaná včetně příkladu. Vždy, než začneme v PHP něco programovat, je dobré se podívat, zda k tomu již nemáme k dispozici nějakou funkci. Jednak si ušetříme práci, vyvarujeme se chybám, které bychom mohli udělat a hlavně vestavěné PHP funkce jsou programované v céčku a jsou tedy z hlediska výkonu úplně někde jinde.

Určitě nemusíte funkce teď zkoumat, jen si přečtěte, co která dělá a když to budete v budoucnu potřebovat, vyhledáte si ji.

array_fill	Naplní pole hodnotami.
array_flip	Otočí klíče a hodnoty v poli.
array_intersect_key	Vrátí pole, jehož klíče jsou průnikem klíčů dvou polí.
array_intersect	Vrátí pole, které je průnikem hodnot zadaných polí.
array_keys	Vrátí pole klíču ze zadaného pole.
array_map	Aplikuje callback (funkci) na všechny prvky v poli.

array_merge	Spojí několik polí do jednoho.
array_pop	Umožňuje používat pole jako zásobník, odebere poslední prvek.
array_push	Umožňuje používat pole jako zásobník, vloží prvek za poslední prvek
array_reverse	Převrátí hodnoty v poli.
array_search	Vyhledá v poli daný prvek.
array_shift	Umožňuje používat pole jako frontu, odebere první prvek.
array_sum	Vrátí součet hodnot v poli.
array_unique	Odstraní duplicitní hodnoty v poli.
array_unshift	Umožňuje pole používat jako frontu, přidá první prvek.
array_values	Vrátí pole všech hodnot z daného pole.
count	Spočítá prvky v poli, případně v objektu.
extract	Rozbalí proměnné z pole do současného scope.
ksort	Seřadí pole podle klíčů.
sort	Seřadí hodnoty v poli od nejmenších po největší.

Pole polí

Pokud vás napadlo, zda mohou být položky pole samy polem, tak to jde. Pole je datový typ jako každý jiný a do pole můžeme vložit i několik polí. Velmi často budete pracovat s polem, jako je toto:

Kostra prikladu

```
$pavel = array(
        'jmeno' => 'Pavel Novák',
        'vek' => '20',
);
$tomas = array(
```

```
        'jmeno' => 'Tomáš Marný',
        'vek' => '50',
);
$jana = array(
        'jmeno' => 'Jana Nová',
        'vek' => '35',
);
$lide = array($pavel, $tomas, $jana);
```

Vytvoříme si 3 pole a tato pole vložíme do jednoho pole. Vnitřní pole představují jednotlivé lidi, vnější je obaluje, abychom s nimi mohli jednoduše hromadně pracovat a třeba je vypsat.

Přesně takovéto pole vám vrátí nějaký databázový dotaz pro výběr uživatelů, jen by asi bylo jméno a příjmení zvlášť. Vypsat byste ho už měli umět, pro jistotu si to uveďme:

```
echo('<table border="1">');
echo('<tr><th>Jméno</th><th>Věk</th></tr>');
foreach ($lide as $clovek)
{
        echo('<tr><td>' . htmlspecialchars($clovek['jmeno']) . '</td>');
        echo('<td>' . htmlspecialchars($clovek['vek']) . '</td></tr>');
}
echo('</table>');
```

Priklad pole_ludia.php

Výstup:

```
<!DOCTYPE html>
<html lang="sk">
    <head>
        <meta charset="UTF-8">
        <title>Pole ludia v PHP</title>
    </head>
    <body>
        <?php

            $pavol = array(
                            'meno' => 'Pavol Novák',
                            'vek' => '20',
                    );
                    $tomas = array(
                            'meno' => 'Tomáš Márn ',
                            'vek' => '50',
                    );
                    $jana = array(
                            'meno' => 'Jana Nová',
```

```
                                'vek' => '35',
                        );
                        $ludia = array($pavol, $tomas, $jana);

                        echo('<table border="1">');
                        echo('<tr><th>Meno</th><th>Vek</th></tr>');
                        foreach ($ludia as $clovek)
                        {
                                echo('<tr><td>' .
htmlspecialchars($clovek['meno']) . '</td>');
                                echo('<td>' .
htmlspecialchars($clovek['vek']) . '</td></tr>');
                        }
                        echo('</table>');

        ?>
    </body>
</html>
```

O takovýchto polích se často mluví jako o vícerozměrných. A to zejména tehdy, když se jedná např. o pole čísel. Takové pole si potom můžeme představit jako tabulku (matici). Pole 3x3 bychom mohli vytvořit např. takto:

```
$matice = array(
        array(1, 2, 3),
        array(4, 5, 6),
        array(7, 8, 9)
);
```

Větší pole bychom samozřejmě vytvářeli cyklem. Vypisovat a pracovat s polem budeme pomocí 2 vnořených for cyklů, jako jsme to dělali u malé násobilky. Ukažme si výpis pole:

Kostra príkladu

```
echo('<table border="1">');
for ($j = 0; $j < 3; $j++)
{
        echo('<tr>');
        for ($i = 0; $i < 3; $i++)
```

```
        {
                echo('<td>' . $matice[$j][$i] . '</td>');
        }
        echo('</tr>');
}
echo('</table>');
```

príklad matica.php

```
<!DOCTYPE html>
<html lang="sk">
    <head>
        <meta  charset="UTF-8">
        <title>2D pole ako matica v PHP</title>
    </head>
    <body>
        <?php


            $matica = array(
                array(1, 2, 3),
                array(4, 5, 6),
                array(7, 8, 9)
            );

            echo('<table border="1">');
            for ($j = 0; $j < 3; $j++)
            {
                echo('<tr>');
                for ($i = 0; $i < 3; $i++)
                {
                    echo('<td>' . $matica[$j][$i] . '</td>');
                }
                echo('</tr>');
            }
            echo('</table>');

        ?>
    </body>
</html>
```

Výsledek:

Prakticky jsem takovéto 2D pole použil např. v online řešiči sudoku. Příště dokončíme úplné základy PHP, uvedeme si funkce pro práci s textovými řetězci a naučíme se deklarovat vlastní funkce. Dnešní příklady jsou jako vždy ke stažení níže.

Cvičení k 12. lekci PHP

Následující 3 cvičení vám pomohou procvičit znalosti programování v PHP z minulé lekce. Ve vlastním zájmu se jepokuste vyřešit sami. Pod článkem máte pro kontrolu řešení ke stažení. Ale pozor, jakmile se na něj podíváte bez vyřešení příkladů, ztrácí pro vás cvičení smysl a nic se nenaučíte 🙂

Pokud si opravdu nebudete vědět rady, podívejte se raději znovu do minulého tutoriálu a pokuste se na to přijít.

Jednoduchý příklad

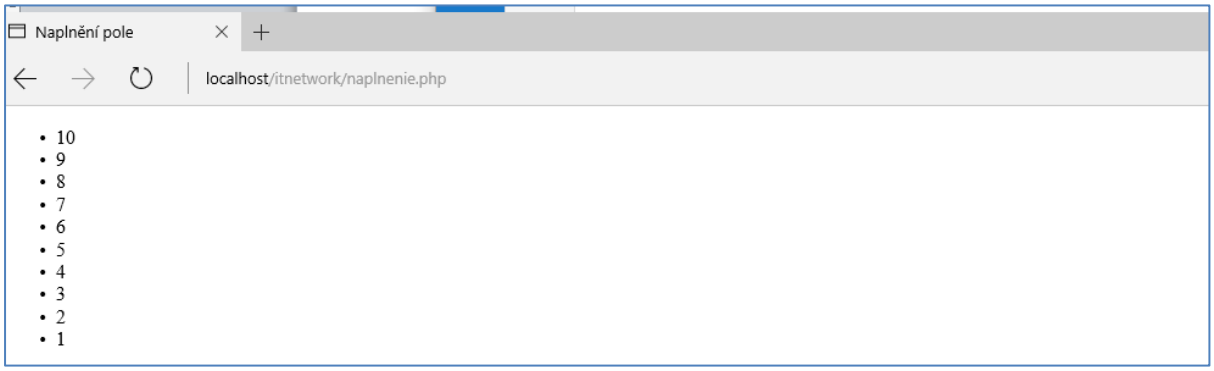
Vytvořte skript, který naplní pole o deseti prvcích postupně čísly od 10 do 1. Pole následně vypište jako seznam pomocí cyklu foreach.

Priklad naplnenie.php

```
<!DOCTYPE html>
<html lang="sk">
<head>
        <meta charset="utf-8" />
        <title>Naplnenie pola</title>
</head>

<body>

<?php

        $cisla = array();
        for ($i = 0; $i < 10; $i++)
        {
                $cisla[$i] = (10 - $i);
```

```
        }

        // Případně můžeme zapsat i takto:
        /*
        for ($i = 10; $i >= 1; $i--)
        {
                $cisla[] = $i;
        }
        */

        echo('<ul>');
        foreach ($cisla as $cislo)
        {
                echo('<li>' . $cislo);
        }
        echo('</ul>');

?>

</body>

</html>
```

Výstup:



Středně pokročilý příklad

Vytvořte skript, kterému zadáte pomocí formuláře plodinu a on zjistí, zda je to ovoce nebo zelenina. Pokud tedy zadáme např. malina, vypíše, že se jedná o ovoce. Pokud brokolice, vypíše, že se jedná o zeleninu. Program samozřejmě rozpozná několik slov a pokud narazí na slovo, které není definované, tak na tuto skutečnost uživatele upozorní. Vyhněte se použití složitého větvení a místo toho použijte pole.

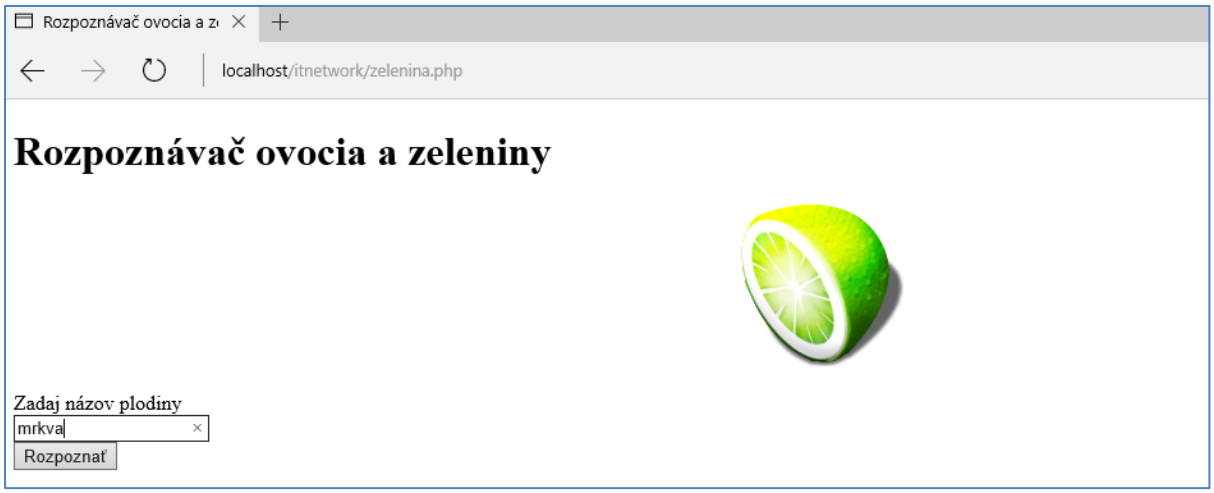
Ukázka obrazovky programu:

Priklad zelenina.php

```
<!DOCTYPE html>
<html lang="sk">
<head>
	<meta charset="utf-8" />
	<title>Rozpoznávač ovocia a zeleniny</title>
</head>

<body>

<h1>Rozpoznávač ovocia a zeleniny</h1>
<p style="text-align: center;">
	<img src="limetka.png" alt="Limetka" />
</p>


<form method="post">
	Zadaj názov plodiny<br />
	<input type="text" name="nazov" /><br />
	<input type="submit" value="Rozpoznať" />
</form>

<?php


	if (isset($_POST['nazov']))
	{
		$plodina = $_POST['nazov'];
		$zeleniny = array("kapusta", "uhorka", "rajcina",
"paprika", "redkovka", "mrkva", "brokolica");
	 $ovocie = array("jablko", "hruška", "pomaranč", "jahoda", "banán",
"kiwi", "malina", "limeta");

		echo('<p>');
		if (array_search($plodina, $ovocie) !== false)
		{
			echo($plodina . " je ovocie");
		}
		elseif (array_search($plodina, $zeleniny) !== false)
		{
			echo($plodina . " je zelenina");
		}
		else
			echo('Tvoju plodinu nepoznám');
		echo('</p>');
	}
```

```
?>

</body>

</html>
```

Výstup:



13. díl - Funkce pro práci s řetězci v PHP

V minulém dílu seriálu tutoriálů se základy PHP jsme si ukázali práci s polem pomocí cyklu. V dnešním dílu řetězce dobereme a ukážeme si PHP funkce, které s nimi pracují.

Textové řetězce a UTF-8

Část PHP funkcí pro práci s řetězci začíná prefixem mb_. Je to z toho důvodu, že tyto funkce podporují UTF-8 kódování (MB jako MultiByte). V kódování UTF-8 se píše naprostá většina webů, protože umí většinu znaků většiny národních abeced. Není tedy problém na webu použít češtinu, dále citovat něco rusky nebo používat speciální znaky jako ☺ ♥. Většina IDE (např. NetBeans) tvoří projekty vždy jako UTF-8. Pokud UTF-8 nepoužíváte, setkáte se časem s velkými problémy, např. tehdy, když budete chtít použít nějakou cizí knihovnu. Její autor totiž určitě počítá s tím, že UTFko používáte.

Jakmile v aplikaci používáme tyto funkce, musíme nejprve nastavit kódování, jinak nebudou korektně fungovat. Kódování stačí nastavit jen jednou v každém skriptu. Pokud se celý váš web zobrazuje přes index, jak jsme si zde ukazovali, stačí nastavení vložit pouze na začátek indexu.

```
mb_internal_encoding("UTF-8");
```

Délka řetězce

Délku řetězce ve znacích získáme pomocí funkce mb_strlen(). Udělejme si malou ukázku, samozřejmě si nad ní vložte ještě řádek výše, který nastaví kódování.

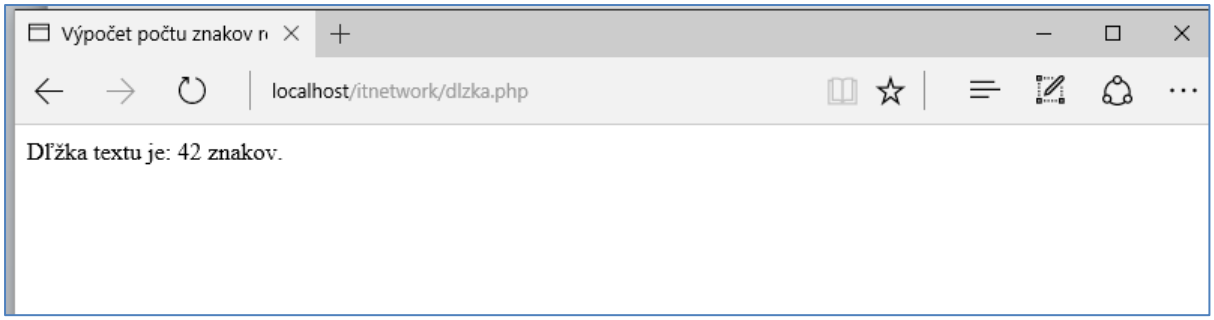
Kostra kódu

```php
$text = "Černé díry jsou tam, kde bůh dělil nulou";
$delka = mb_strlen($text);
echo("Délka textu je $delka znaků.");
```

Priklad dlzka.php

```php
<!DOCTYPE html>
<html lang="sk">
    <head>
        <meta charset="UTF-8">
        <title>Výpočet počtu znakov reťazca</title>
    </head>
    <body>
        <?php
        $text = "Čierne diery sú tam, kde Boh delil nulou";
        $dlzka = mb_strlen($text);
        echo("Dĺžka textu je: $dlzka znakov.");

        ?>
    </body>
</html>
```

Výstup:



Pozn.: V zastaralých učebnicích a tutoriálech naleznete použití funkcí bez prefixu mb_. Tedy místo mb_strlen() jen strlen(). Tyto funkce nikdy nepoužívejte, jelikož neumí UTF-8 a budou vám vracet špatný výstup.

Např. "Č" je v UTF kódování uloženo jako 2 znaky (2 bajty, jako háček a c). Funkce s prefixem mb_ bere č jako jeden znak, funkce bez tohoto prefixu ho bere jako 2 znaky. Vrací tedy špatně délku řetězců s diakritikou a nedokáže rozeznat o která písmena se jedná. PHP obsahuje z důvodu zpětné kompatibility mnoho funkcí, které UTF kódování nepodporují, měli byste se vždy podívat, zda je funkce tzv. multibyte-safe a případně najít její multibyte variantu.

Práce s podřetězci

Určitému úseku řetězce říkáme podřetězec. Ukažme si nějaké příklady s podřetězci, jelikož s těmi budeme často pracovat.

Zjištění pozice podřetězce

Pokud chceme zjistit, na jaké pozici se v řetězci nachází konkrétní podřetězec nebo zda ho text vůbec obsahuje, použijeme funkci mb_strpos(). Abychom si to udělali zajímavější, budeme chtít, aby nám nezáleželo na velikosti písmen. Z toho důvodu nejprve celý řetězec převedeme na velká písmena pomocí funkce mb_strtoupper() a poté v něm budeme hledat podřetězec, též velkými písmeny.

Kostra scriptu

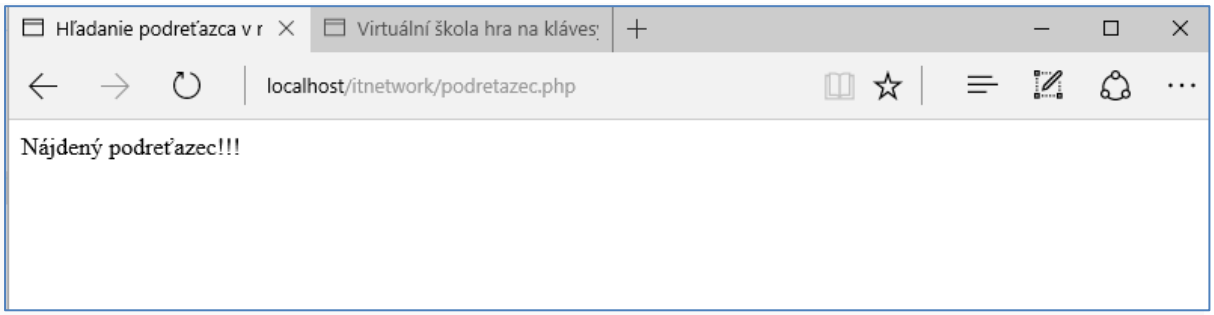
```
$retezec = mb_strtoupper('PHP tutoriály na Devbooku.');
$podretezec = mb_strtoupper('devbook');

if (mb_strpos($retezec, $podretezec) !== false)
        echo "Nalezeno";
else
        echo("Nenalezeno");
```

Príklad podretazec.php

```
<!DOCTYPE html>
<html lang="sk">
    <head>
        <meta charset="UTF-8">
        <title>Hľadanie podreťazca v reťazci</title>
    </head>
    <body>
        <?php
        $retazec = mb_strtoupper('PHP tutoriály na Devbooku.');
$podretazec = mb_strtoupper('devbook');

if (mb_strpos($retazec, $podretazec) !== false)
        echo "Nájdený podreťazec!!!";
else
```

```
        echo("Nenájdený podreťazec:devbook");

        ?>
    </body>
</html>
```

Výstup:



mb_strpos() vrací 0 v případě, že je podřetězec na první pozici a false v případě, že nebyl nalezen. Z tohoto důvodu je nutné výsledek porovnávat i s ohledem na datový typ, jak jsme se to učili u podmínek. Jinak by bylo false a 0 vyhodnoceno stejně a podřetězec by nebyl nalezen v případě, že by jím řetězec začínal.

K funkci mb_strpos() existuje ještě funkce mb_strrpos() (r navíc jako reverse), která funguje úplně stejně, jen vyhledává od konce řetězce. Hodí se např. když zjišťujeme příponu souboru.

V problematice vyhledávání podřetězců se řetězci často říká kupka sena (haystack) a podřetězci jehla (needle).

Získání podřetězce podle pozice

Podřetězec získáme pomocí funkce mb_substr(), která bere v parametrech řetězec, index, od kterého podřetězec začíná a délku podřetězce. Zkusme si to:

Kostra scriptu

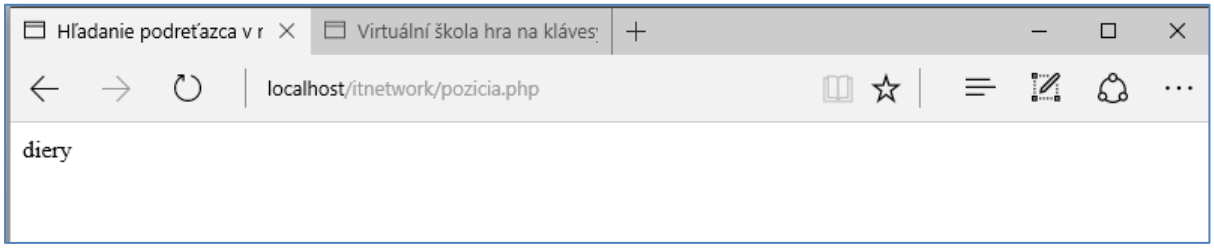
```
$text = "Černé díry jsou tam, kde bůh dělil nulou.";
echo mb_substr($text, 6, 4);
```

Príklad pozicia.php

```
<!DOCTYPE html>
<html lang="sk">
```

```
    <head>
        <meta charset="UTF-8">
        <title>Hľadanie podreťazca v reťazci</title>
    </head>
    <body>
        <?php
       $text = "Čierne diery sú tam , kde Boh delil nulou.";
        echo mb_substr($text, 7, 6);
        ?>
    </body>
</html>
```

Výstup:



Získali jsme podřetězec od 7. znaku, dlouhý 6 znaky.

Přístup k určitému znaku

S textovými řetězci lze v novějších verzích PHP pracovat jako s polem a to tímto způsobem:

Kostra scriptu

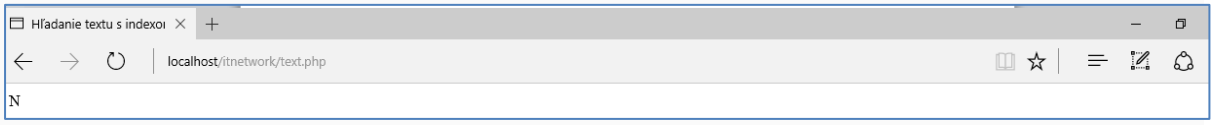
```
$text = "Nějaký text";
echo $text[0];
```

Priklad text.php

```
<!DOCTYPE html>
<html lang="sk">
    <head>
        <meta charset="UTF-8">
        <title>Hľadanie textu s indexom</title>
    </head>
    <body>
        <?php
       $text = "Neaký text";
        echo $text[0];
       ?>
```

```
    </body>
</html>
```

Výstup



Podobného výsledku šlo docílit v minulosti i pomocí složených závorek, ale tato syntaxe byla z PHP odstraněna. Kód výše vypíše 1. znak. **Bohužel tento způsob ještě nepodporuje Unicode (UTF-8) a proto ho nepoužívejte**. Pokud potřebujete přistoupit k nějakému znaku, jednoduše ho zkopírujte jako podřetězec pomocí výše zmíněné funkce mb_substr().

Nahrazení podřetězce

V textu můžeme velmi jednoduše nahradit nějaký podřetězec jiným. Docílíme toho použitím funkce str_replace a můžeme tak jednoduše zabezpečit např. emailovou adresu před spamboty tak, že znak zaviináče nahradíme textem "(zavináč)". Roboti pak nepoznají, že se jedná o email a nebudou vám nabízet výhodné půjčky

Kostra scriptu

```
$adresa = 'devbook@devbook.cz';
$osetrenaAdresa = str_replace('@', '(zavináč)', $adresa);
echo $osetrenaAdresa;
```

Priklad zavinac.php

```
<!DOCTYPE html>
<html lang="sk">
    <head>
        <meta charset="UTF-8">
        <title>Nahradenie textu v reťazci iným reťazcom.</title>
    </head>
    <body>
        <?php
       $adresa = 'devbook@devbook.cz';
        $osetrenaAdresa = str_replace('@', '(zavináč)', $adresa);
    echo $osetrenaAdresa;

       ?>
    </body>
</html>
```

Výstup:

Pokud by v textu bylo více takových podřetězců, funkce nahradí všechny.

Nahrazení podle slovníku

Pokud potřebujeme provést více nahrazení, PHP nabízí funkci strtr (jako STRing TRanslate, ano, pojmenování mnoha funkcí je v PHP velmi zavádějící). Funkce bere v parametrech řetězec a slovník, kde jsou jako klíče podřetězce, které chceme nahradit a jako hodnoty řetězce, kterými je chceme nahradit.

Funkce se často používá k nahrazení textových smajlíků v nějakém textu za HTML obrázky. Zkusme si to:

```
$slovnik = array(
        ':)' => '<img src="usmev.png" alt="úsměv" />',
        ':D' => '<img src="smich.png" alt="smích" />',
);
echo strtr('Ahoj :) Je mi fajn, protože jsem objevil devbook :D',
$slovnik);
```

Výstup:

```
Ahoj <img src="usmev.png" alt="úsměv" /> Je mi fajn, protože jsem objevil
devbook <img src="smich.png" alt="smích" />'
```

Rozdělení řetězce na pole podřetězců

Velmi užitečnou dvojicí funkcí jsou expresivně pojmenované explode() a implode(). Explode() rozdělí řetězec na pole podřetězců pomocí určitého oddělovače. Implode() naopak spojí podřetězce v poli do jednoho dlouhého řetězce a mezi podřetězce vloží oddělovač. Oddělovači se někdy expresivně říká lepidlo.

Níže uvedený jednoduchý program bere na vstupu řetězec s několika čísly, které jsou oddělené čárkou. Z těchto čísel následně vypočítá součet.

```
$vstup = "1,5,87,65,42,4,456,8,5,98,54,89";
$cisla = explode(',', $vstup);
echo array_sum($cisla);
```

Explode() rozdělí řetězec podle čárky a vrátí pole jeho částí. Pomocí funkce array_sum() následně získáme součet prvků v poli. Pokud přicházíte z nějakého nízkého jazyka, asi se

divíte, jak je v PHP vše jednoduché. Je to z toho důvodu, že PHP je tzv. vysoký jazyk. Právě díky tomu můžeme svou energii zaměřit na vývoj aplikace a ne na řešení základních problémů.

PHP funkce pro práci s řetězci

Na závěr si uveďme seznam těch nejdůležitějších funkcí, které nám PHP pro práci s řetězci nabízí. Každou si můžete rozkliknout a podívat se, jak se používá. Opět je nemusíte umět nazpaměť, stačí vědět, že tam jsou a že si je v případě potřeby můžete vyhledat.

mb_internal_encoding	Nastavení kódování.
mb_strlen	Získá délku řetězce.
mb_strpos	Najde pozici prvního výskytu podřetězce v řetězci.
mb_substr	Vrátí podřetězec od startovní pozice s určitým počtem znaků.
mb_strtoupper	Převede všechna písmena v rětězci na velká.
mb_strtolower	Převede všechna písmena v rětězci na malá.
trim	Odstraní bílé místo na okolo řetězce.
htmlspecialchars	Převede speciální znaky v textu na HTML entity.
htmlspecialchars_decode	Převede entity v textu zpět na speciální znaky.
strip_tags	Odstraní z daného řětězce HTML tagy.
nl2br	Nahradí konce řádků (\n) tagem
str_replace	Nahradí všechny výskyty podřetězce v řetězci daným podřetězcem
strtr	Přeloží podřetězce podle slovníku.
parse_str	Rozbalí proměnné z textového řetězce ve tvaru QUERY stringu.
explode	Převede řetězec na pole podřetězců.

implode	Zabalí pole do textového řetězce.
hash	Vypočítá otisk (hash) řetězce.

V příštím dílu si ukážeme, jak v PHP deklarovat vlastní funkce.

14. díl - Tvorba a použitie funkcií v PHP

Funkcia je uložená sada príkazov, ktorá má svoje meno a môžeme ju vykonávať.

Jazyk PHP, tak ako aj iné programovacie jazyky, umožňuje vytvárať funkcie viacerými spôsobmi. Na rozdiel od iných programovacích jazykov ponúka aj viac možností pre prácu s funkciami

.Funkcia je blok príkazov, ktoré autor má v úmysle použiť v programe viac krát. Funkcia sa skladá z hlavičky a tela. Hlavička funkcie sa skladá z kľúčového slova `function`, názvu funkcie a oblých zátvoriek `()`, ktoré môžu obsahovať parametre. Parametre sú lokálnymi premennými funkcie. Telo funkcie je tvorené zloženými zátvorkami `{}`, ktoré definujú programový blok, a príkazmi tvoriacimi telo funkcie. Funkcia sa spúšťa jej volaním.

```
function functionName() {
    code to be executed;
}
```

Pre spustenie funkcie je potrebné zadať názov funkcie a oblé zátvorky `()`. V nich môžu byť umiestnené argumenty, ktoré budú predané funkcii na spracovanie. Počet argumentov musí byť rovnaký, prípadne väčší ako počet parametrov, inak vznikne chyba. To neplatí, ak hlavička funkcie obsahuje nepovinné parametre. PHP ich samo o sebe ponúka obrovské množstvo.

PHP obsahuje veľké množstvo vstavaných funkcií - napr. na prácu s reťazcami, časom, súbormi, databázou; matematické, sieťové funkcie a podobne.

Niektoré sú jeho neoddeliteľnou súčasťou, kvôli iným je nutné pridať do PHP *knižnicu*. Nejaké knižnice sú k PHP tiež pribalené.

1. Matematické funkcie

abs - Absolútna hodnota

cos - Cosinus

max - Nájdenie najväčšie hodnoty

min - Nájdenie najmenšie hodnoty

pi - Získanie hodnoty pí

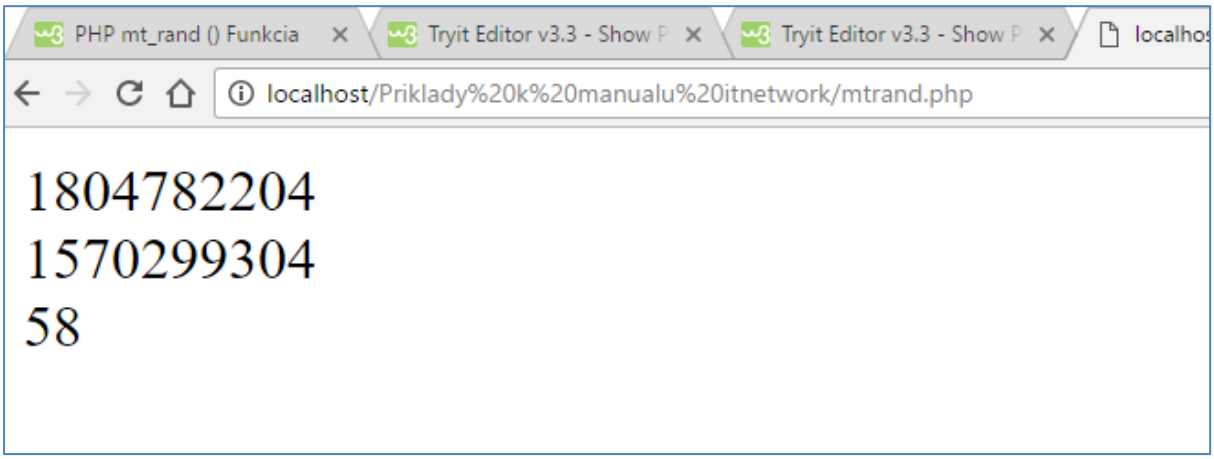
rand - Generovanie náhodného čísla

priklad na použitie funkcie generovania náhodných čísel

generator.php

```
<!DOCTYPE html>
<html>
<body>

<?php
echo(mt_rand() . "<br>");/*Generátor náhodných čísel*/
echo(mt_rand() . "<br>");
echo(mt_rand(10,100));/*Generátor nahodných čísel z rozsahu 10 až 100*/
?>

</body>
</html>
```

Server php



Funkcia rand () generuje náhodné číslo.

Tip: Ak chcete náhodné číslo medzi 10 a 100 (vrátane), použite rand (10,100).

Tip: mt_rand () funkcia vytvorí lepšiu náhodnú hodnotu, a je 4 krát rýchlejší

ako rand ().

Priklad sportka

Loto.php

```
<!DOCTYPE html>
<html>
    <head>
    <meta charset="UTF-8">
       <title>Generovanie čísel Športky</title>
    </head>
    <body style ="background-color:green" >
    <h2>Vylosované čísla športky</h2>
          <?php


             for ($i = 1; $i <=7; $i++)
              {
                    echo('<br>');
                    for ($j = 1; $j <=7; $j++)
                     {
                         $vysledok = rand ( 1 , 49 );
                  echo "<span style='color:red;'>".$vysledok."  "."</span>";

                     }
             }

             ?>
    </body>
</html>
```

Server php

round - Zaokrúhlenie čísla

sqrt - Odmocnina

tan – Tangens

2. Reťazcové funkcie

addslashes - Opatriť reťazec lomítkami

chr - Vrátiť určitý znak

explode - Rozdeľuje reťazec iným reťazcom

implode - Spoj prvky poľa pomocou reťazca

join - Spoj prvky poľa pomocou reťazca

md5 - Spočítať MD5 hash reťazca

ord - Vrátiť ASCII hodnotu znaku

strchr - Nájsť prvý výskyt znaku

strlen - Zistiť dĺžku reťazca

strpos - Nájsť pozíciu prvého výskytu reťazca

strrchr - Nájsť posledný výskyt znaku v reťazci

strrev - Obrátiť reťazec

strstr - Nájsť prvý výskyt reťazca

strtolower - Zmeniť reťazec na malé písmená

strtoupper - Zmeniť reťazec na veľké písmená

substr_replace - Nahradiť časť reťazca iným reťazcom

substr - Vrátiť časť reťazca

Príklad na funkciu vyhľadania podreťazca v reťazci

Získanie podreťazca podľa pozície, získame pomocou funkcie mb_substr (), ktorá berie v parametroch:

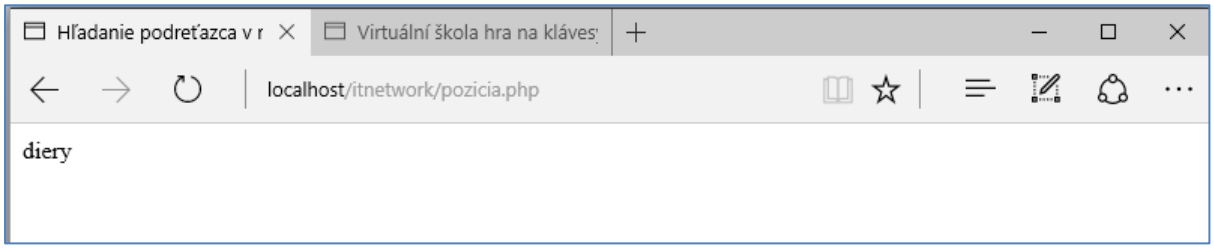
- reťazec,

- index, od ktorého podreťazec začína
- a dĺžku podreťazca. Skúsme si to:

Príklad pozicia.php

```
<!DOCTYPE html>
<html lang="sk">
    <head>
        <meta charset="UTF-8">
        <title>Hľadanie podreťazca v reťazci</title>
    </head>
    <body>
        <?php
       $text = "Čierne diery sú tam , kde Boh delil nulou.";
        echo mb_substr($text, 7, 6);
        ?>
    </body>
</html>
```

Výstup:



Získali jsme podreťazec od 7. znaku, dlhý 6 znakov.

ucfirst - Zmení prvé písmeno reťazca na veľké

ucwords - Zmeniť prvý znak každého slova v reťazci na veľké písmeno

3. Funkcie pre prácu s poľom

array_merge - Zlúčiť dve alebo viac polí

array_push - Pridať jeden alebo viac prvkov na koniec poľa

array_rand - Vybrať náhodne jeden alebo viac prvkov poľa

array_reverse - Vrátiť pole s prvkami v opačnom poradí

array_shift - Odstrániť prvok zo začiatku poľa

array_splice - Odstrániť časť poľa a nahradiť ju niečím iným

array_unique - Odstrániť z poľa duplicitné hodnoty

array_values - Vrátiť všetky hodnoty v poli

array - Vytvoriť pole

end - Nastaviť vnútorný ukazovateľ poľa na jeho posledný prvok

foreach - Posunúť interný ukazovateľ poľa

pos - Získať súčasný prvok poľa

reset - Nastaviť interný ukazovateľ poľa na jeho prvý prvok

rsort - Zoradiť pole zostupne

shuffle - Zamiešať pole

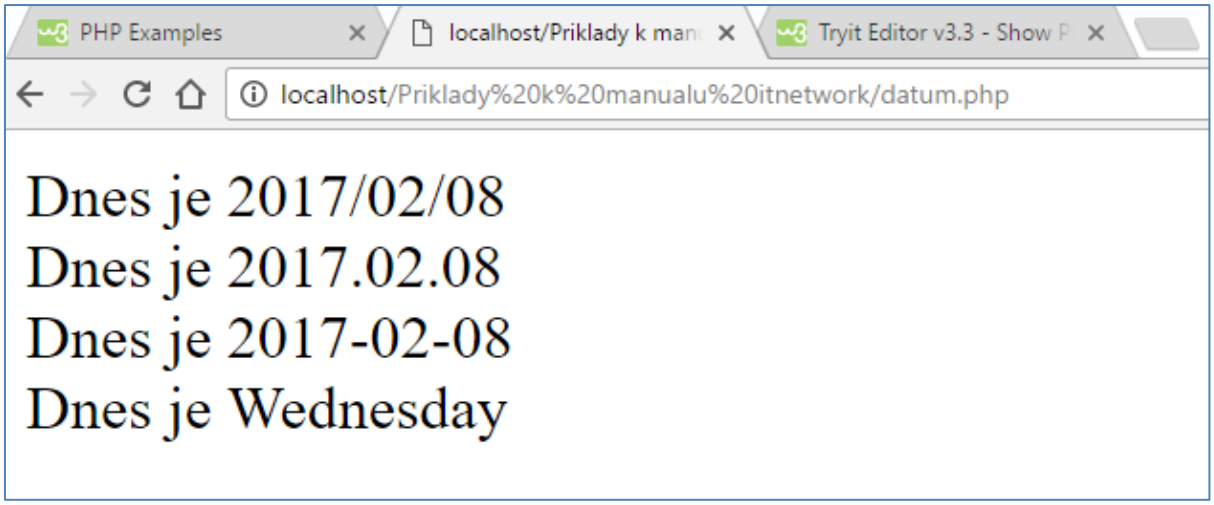
sizeof - Zistiť počet prvkov v poli

sort - Zoradiť pole

4Časové funkcie-Priklad použitia funkcie date

Datum.php

```
<!DOCTYPE html>
<html>
<body>

<?php
echo "Dnes je " . date("Y/m/d") . "<br>";
echo "Dnes je " . date("Y.m.d") . "<br>";
echo "Dnes je " . date("Y-m-d") . "<br>";
echo "Dnes je " . date("l");
?>

</body>
</html>
```

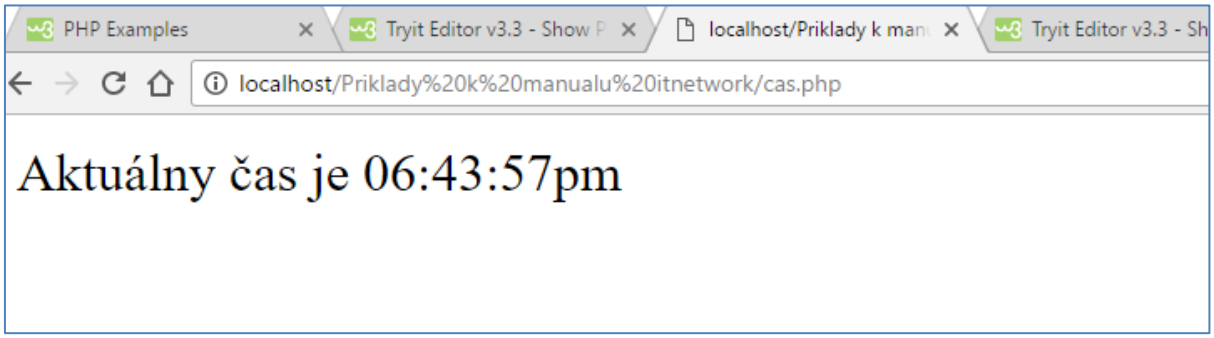


Priklad na použitie funkcie TIME

Cas.php

```
<!DOCTYPE html>
<html>
<body>

<?php
echo "Aktuálny čas je " . date("h:i:sa");
?>

</body>
</html>
```

Php server



Užívateľské funkcie

Okrem toho si môžeme vytvoriť vlastné funkcie.

Odkaz na itnetwork

Syntax uživateľom definovanej funkcie

```
function functionName() {
    code to be executed;
}
```

Poznámka: Názov funkcie môže začínať písmenom alebo podčiarknikom (nie číslom).

Tip: Dajte Funkcií názov, ktorý odráža to, čo funkcia robí!

1.Funkcia bez parametrov

Pozdrav1.php

```
<!DOCTYPE html>
<html>
<body>

<?php
function pozdrav() {
    echo ("Ahoj všetci!");
}

pozdrav();
?>

</body>

</html>
```

2.Funkcia s parametrom

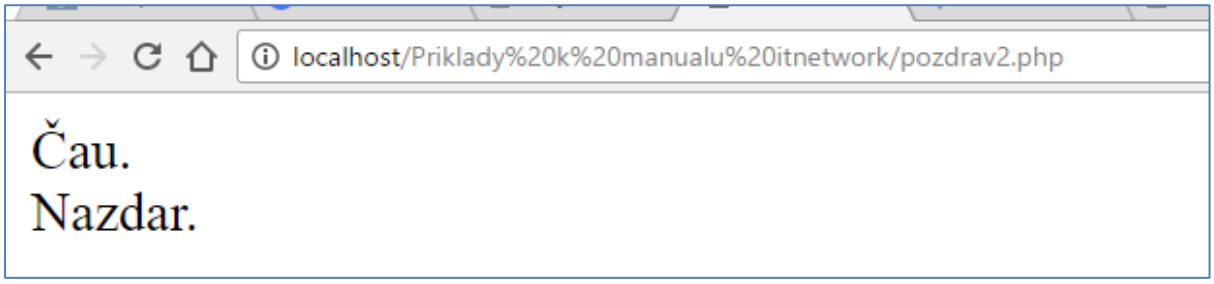
Chceme, aby naša funkcia podporovala rôzne pozdravy. Využijeme tzv. parameter funkcie - ten určuje, ako sa funkcia bude správať. Píšeme ho do zátvoriek.

Pozdrav2.php

```
<!DOCTYPE html>
<html>
<body>

<?php
```

```
function pozdrav($text) {
    echo ("$text.<br>");
}

pozdrav("Čau");/* volanie funkcie s parametrom Čau */
pozdrav("Nazdar");/* volanie funkcie s parametrom Nazdar */
?>

</body>

</html>
```

Php server



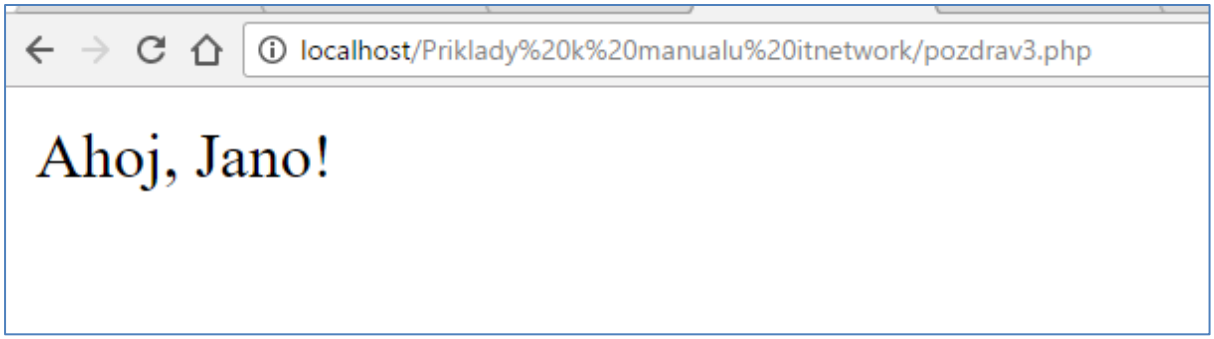
3.Funkcia obsahujúca viac parametrov

Parametrov môže byť aj viac - vtedy ich oddeľujeme čiarkami.

Pozdrav3.php

```
<!DOCTYPE html>
<html>
<body>

<?php
        function pozdrav($text, $meno) {
                        echo ("$text, $meno!");
                        }

        /* volanie funkcie s 2 parametrami */
        pozdrav("Ahoj", "Jano"); /* vypíše: Ahoj, Jano!*/
?>
</body>

</html>
```

Server php

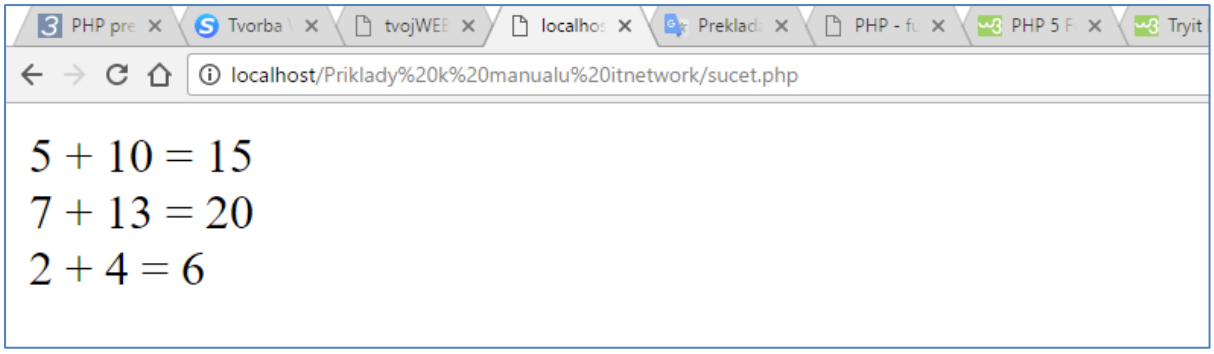


4.Funkcie vracajúce hodnotu

Sucet.php

```
<!DOCTYPE html>
<html>
<body>

<?php
function sum($x, $y) {
    $z = $x + $y;
    return $z;
}

echo ("5 + 10 = " . sum(5,10) . "<br>");
echo ("7 + 13 = " . sum(7,13) . "<br>");
echo ("2 + 4 = " . sum(2,4));
?>

</body>
</html>
```

Server php



Faktorial.php

```php
<?php
function fak($x)
{
        if ($x>0) return $x * fak($x-1);
        else return 1;
}
echo ("Faktorial 5! je".fak(5)."<br>");
echo ("Faktorial 4! je".fak(4)."<br>");
echo ("Faktorial 3! je".fak(3)."<br>");
echo ("Faktorial 2! je".fak(2)."<br>");
echo ("Faktorial 1! je".fak(1)."<br>");
?>
```

Php server

formulár

Väčšina ľudí sa dnes stretlo s formulárom na webe. Najčastejšie to býva napríklad pri zakladaní freemailové poštovej schránky na niektorom zo serverov, ktoré túto službu poskytujú. Do formulára užívateľ vyplní požadované údaje a pokúsi sa formulár odoslať (väčšinou kliknutím na rovnomennej tlačidlo). Po kliknutí na toto tlačidlo sa zadané informácie odošlú na server a tam sa spravidla uskutoční ich kontrola (či je vyplnené všetko potrebné, či je to vyplnené aspoň trochu zmysluplne - v rámci

možností počítačového spracovania) a buď sa formulár zobrazí znova s patričnými inštrukciami a súpisom chybne zadaných údajov alebo dôjde k spracovaniu zadaných dát a napríklad ich uloženie do databázy na server.

Tvorba webových formulárov pozostáva z dvoch krokov:

- Návrh štruktúry formuláre a vytvorenie jeho HTML kódu.
- Implementácia funkčnosti formulára (kontrola, vyhodnotenie a spracovanie zadaných dát).

Vloženie formulára do HTML stránky

Formulár sa do stránky vkladá medzi značky **<form>** a **</ form>** . Ako atribúty tagu form sa udávajú nasledujúce dva:

- **action** - akcia, ktorá sa má po odoslaní formulára vykonať. Ide o URL (meno skriptu) skriptu, ktorému sa majú zadané informácie z formulára poslať.
- **method** - metóda, ktorou sa budú prenášať informácie z formulára na server. Možné varianty sú **get** alebo **post** .

ŠTRUKTÚRA FORMULÁRA

```
<form action="zpracujformular.php" method="get">
obsah formuláře (množina textových polí, přepínačů,...)
</form>
```

Stručne k metódam POST a GET

Atribút method značky form určuje spôsob, ktorým sa budú z formulára odosielať dáta. Ktorú metódu zvolíme záleží na nás, ale musíme uvažovať nasledujúce:

- **get** - všetky dáta z formulára sa posielajú cez URL (adresu viditeľnú v prehliadači) stránky a vloží sa do riadku s adresou za meno stránky oddelené otáznikom. Túto metódu je možné použiť kdekoľvek, ale väčšinou sa používa u jednoduchších formulárov, kde nedochádza k posielanie väčšieho množstva dát. Get sa neodporúča používať kdekoľvek to naozaj nie je nutné, pretože ktokoľvek VŠETKA posielané dáta vidí a môže ich teda ľubovoľne meniť. Toto predstavuje bezpečnostné riziko. Naviac dĺžka URL je

obmedzená (pozri napríklad citácia z webu Microsoftu): *Microsoft Internet Explorer má maximálnu dĺžku adresy URL (Uniform Resource Locator) 2 083 znakov. Rovnako maximálna dĺžka cesty je v tejto aplikácii obmedzená na 2 048 znakov. Toto obmedzenie platí pre adresy URL požiadaviek POST aj požiadaviek GET. V prípade použitia metódy GET ste obmedzený maximálnou hodnotou 2 048 znakov mínus počet znakov skutočnej cesty. Metóda POST však nie je obmedzená veľkosťou adresy URL pre odosielanie dvojíc názvov a hodnôt. Tieto dvojice sú prenášané v hlavičke, nie v adrese URL.* Mohlo by sa zdať, že je takýto počet znakov dostatočný, ale napríklad v prípade formulárov s dlhými textovými reťazcami (predstavte si dotazník s 30 otázkami, kde na každú otázku môže užívateľ napísať odpoveď v dĺžke až 256 znakov) alebo u komplexných webových aplikácií (ktoré pracujú s veľkým množstvom parametrov) môže byť tento počet už obmedzujúce a narúšajúce vôbec možnosť danú aplikáciu prevádzkovať. Zostáva zodpovedať otázku, čo sa stane, keď dĺžku url prekročíte. Dôjde k tomu, že sa vezme maximálna možná dĺžka URL a zvyšok nad povolenú dĺžku sa jednoducho zahodí (reťazec sa sprava oreže).

- **post (http://cs.wikipedia.org/wiki/POST) -** dáta neposiela v URL, ale ako samstatný HTTP objekt. Jej jedinou nevýhodou je fakt, že pri obnovení stránky (refresh) zobrazí tabuľku s otázkou, či si POST dáta prajeme naozaj znova odoslať, čo môže byť pre užívateľov mätúce (často sa preto metóda post nepoužíva v rámci firemných intranetových stránok, aby neboli užívatelia frustrovaní a netelefonoval zbytočne na príslušné oddelenia s otázkou, či zobrazený dialóg môžu potvrdiť).

1.Textové pole - text

vlozeny-text

```
<input name="meno-textoveho-pole" type="text" value="vlozeny-text"
size="20" />
```

PR.textového poľa

```
                <input name="poleMeno" type="text" value="" size="20">
```

Pole.php

```
<!DOCTYPE html>
<html lang="sk">
    <head>
        <meta charset="UTF-8">
```

```
        <title>Faktorial čísla</title>
    </head>
    <body>
        <p>Vitajte v kalkulačke, zadajte číslo a získajte jeho
faktorial.</p>

        <form method="POST" action="faktorial.php">
            <input name="cislo" type="number" /><br /><br>
            <input type="submit" value="Vypočítaj !!!" />
        </form>

    </body>
</html>
```

faktorial.php

```
<!DOCTYPE html>
<html lang="sk">
    <head>
        <meta charset="UTF-8">
        <title>Faktorial čísla</title>
    </head>
    <body>
        <?php
        $suma = 0;
        $pocitadlo = 1;
            $fakt = $_POST['cislo'] ;
            while()
            echo("Faktorial čisla:$fakt je $suma");
        ?>
    </body>
</html>
```

2.Textové pole pre dlhší text - textarea

```
<textarea name="textove-pole-vetsi" cols="30" rows="10">

Priklad na textové pole
```

```
<textarea name="textovePoleVacsie" cols="30" rows="5" />Sem uvedte
komentár</textarea><br><br>
```

3. Zaškrtávacie pole - checkbox

A

B

```
<input name="zaskrtavacie-poleA" type="checkbox" value="zaskrtnutie"
checked="checked" />A
<input name="zaskrtavacie-poleB" type="checkbox" value="nezaskrtnute" />B
```

Priklad na zaškrtávacie pole

```
<input name="poleA" type="checkbox" value="Female" checked="checked" />Žena
<input name="poleB" type="checkbox" value="Male" />Muž
```

4.Výberové pole – radio

A

B

C

```
<input name="vyberove-pole" type="radio" value="A" checked="checked" />A
<input name="vyberove-pole" type="radio" value="B" />B
<input name="vyberove-pole" type="radio" value="C" />C
```

priklad na radio

```
<input name="vyberoveA" type="radio" value="A" />Automobily
<input name="vyberoveB" type="radio" value="B" />Móda
<input name="vyberoveC" type="radio" value="C" />Počítače
```

5.Zoznam s výberom možností - select

B

```
<select name="zoznam">
<option value="A">A</option>
<option value="B" selected="selected">B</option>
<option value="C">C</option>
</select>
```

priklad na select

```
<select name="zoznam">
   <option value="A">HTML</option>
   <option value="B" selected="selected">CSS</option>
   <option value="C">PHP</option>
</select><br><br>
```

6.Tlačidlo pre odoslanie formulára - submit

Nakoniec ešte zmienime tlačidlo pre odoslanie formulára. Toto tlačidlo musí byť vo formulári prítomných, aby sa vôbec dáta z formulára niekam odoslala (po stlačení tohto tlačidla v tejto stránke sa odošle formulár obsahujúci iba toto tlačidlo štandardnou metódou get súčasne zobrazenej stránke).

odišli formulár

```
<input name="odslat-formular" type="submit" value="Odošli formulá />
```

priklad na odosielacie tlačidlo

```
            <input type="submit" value="Odošli !!!" />
```

priklad Formulára

Registračný formulár-zadanie

Zostavte jednoduchý HTML formulár, ktorý bude slúžiť pre registráciu užívateľa povedzme na nejaký server diskusných. Formulár bude obsahovať nasledujúce údaje:

- Meno
- priezvisko
- E-mail
- Pohlavie
- kometárov
- Témy, ktoré používateľa zaujímajú

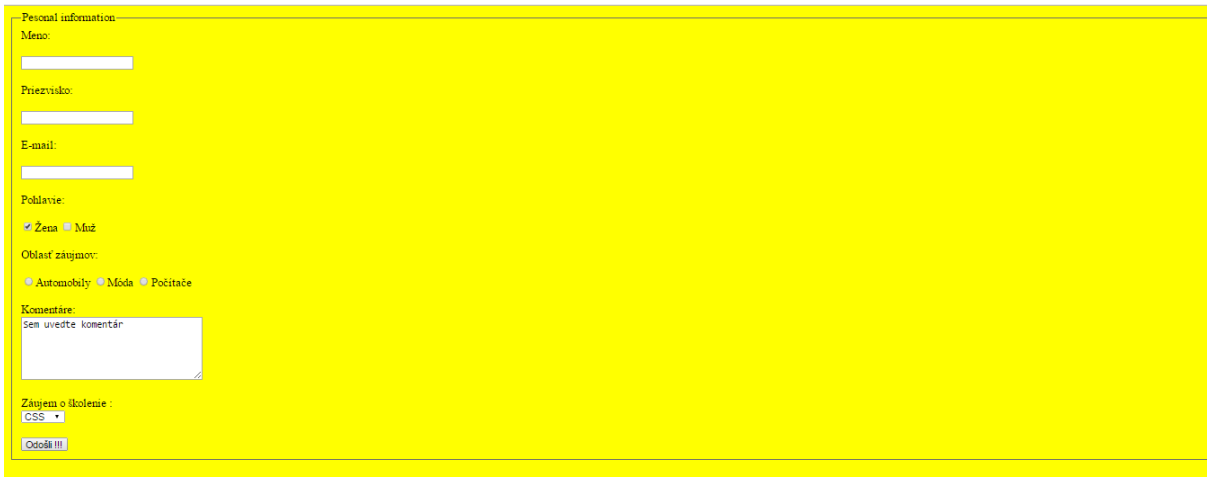
Ešte treba dolniť súbor,ktorý tirto ôdaje spracuje.

formular.html

```
<!DOCTYPE html>
<html lang="sk">
    <head>
        <meta charset="UTF-8">
        <title>Kontaktný formulár</title>
    </head>
    <body style = "background-color: yellow";>
            <form method="POST" action="formular.php">
            <fieldset>
            <legend> Pesonal information </legend>
            Meno:<br><br>
            <input name="poleMeno" type="text" value="" size="20"
/><br><br>
```

```
             Priezvisko:<br><br>
            <input name="polePriezvisko" type="text" value="" size="20"
/><br><br>
             E-mail:<br><br>
            <input name="poleEmail" type="text" value="" size="20"
/><br><br>
            Pohlavie:<br><br>
            <input name="poleA" type="checkbox" value="Female"
checked="checked" />Žena
            <input name="poleB" type="checkbox" value="Male" />Muž <br><br>
            Oblasť záujmov:<br><br>
            <input name="vyberoveA" type="radio" value="A" />Automobily
            <input name="vyberoveB" type="radio" value="B" />Móda
            <input name="vyberoveC" type="radio" value="C"
/>Počítače<br><br>
            Komentáre:<br>
            <textarea name="textovePoleVacsie" cols="30" rows="5" />Sem
uvedte komentár</textarea><br><br>
            Záujem o školenie :<br>
             <select name="zoznam">
                <option value="A">HTML</option>
                <option value="B" selected="selected">CSS</option>
                <option value="C">PHP</option>
             </select><br><br>
            <input type="submit" value="Odošli !!!" />
        </fieldset>
        </form>
        </body>
</html>
```

chrome:



Vyhodnotenie časti formulara:

Pr.formular1.html

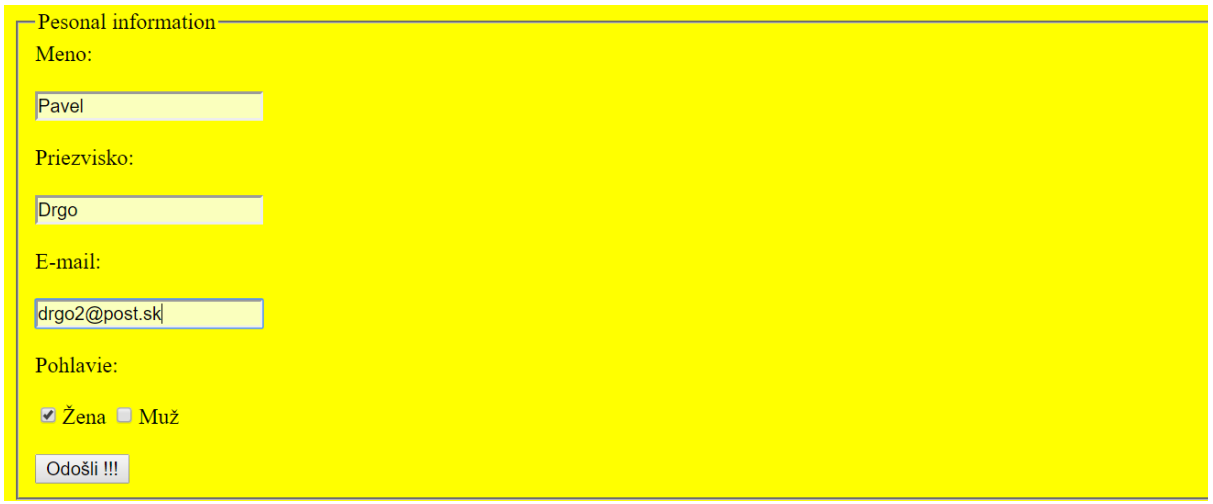
```
<!DOCTYPE html>
<html lang="sk">
    <head>
        <meta charset="UTF-8">
        <title>Kontaktný formulár</title>
    </head>
    <body style = "background-color: yellow";>
            <form method="POST" action="formular1.php">
            <fieldset>
            <legend> Pesonal information </legend>
            Meno:<br><br>
            <input name="poleMeno" type="text" value="" size="20"
/><br><br>
             Priezvisko:<br><br>
            <input name="polePriezvisko" type="text" value="" size="20"
/><br><br>
             E-mail:<br><br>
            <input name="poleEmail" type="text" value="" size="20"
/><br><br>
            Pohlavie:<br><br>
            <input name="pohlavie" type="checkbox" value="Female"
checked="checked" />Žena
            <input name="pohlavie" type="checkbox" value="Male" />Muž
<br><br>
            <input type="submit" value="Odošli !!!" />
        </fieldset>
        </form>
        </body>
</html>
```

Formular1.php

```
<!DOCTYPE html>
<html lang="sk">
    <head>
        <meta charset="UTF-8">
        <title>Dotazník</title>
    </head>
    <body>
        <?php
             $meno = $_POST['poleMeno'] ;
             $priezvisko = $_POST['polePriezvisko'] ;
             $email = $_POST['poleEmail'] ;
```

```
            $pohlavie = $_POST['pohlavie'] ;
            echo("Tvoje meno je:".$meno."<br><br>");
           echo("Tvoje priezvisko je:".$priezvisko."<br><br>");
           echo("Tvoja emailova adresa je:".$email."<br><br>");
           if ($pohlavie =="Male")
           echo "Si muž";
           else
           echo " Sorry, ale si žena";
        ?>
    </body>
</html>
```

Server php



Tvoje meno je:Pavel

Tvoje priezvisko je:Drgo

Tvoja emailova adresa je:drgo2@post.sk

Sorry, ale si žena

Cvičenie generátor náhodných čísel

```
<!DOCTYPE html>
<html lang="sk">
<head>
        <meta charset="utf-8" />
```

```
        <title>Generátor čísel športky</title>
</head>

<body>
<h2 align="center">Generátor čísel športky</h2>
        <?php


                echo('<table border="1">');
                for ($i = 0; $i < 7; $i++)
                {
                        echo('<tr>');

                        for ($j = 0; $j < 7; $j++)
                        {

                        $styl = 'style="background: blue;';
                        $vysledok = rand ( 1 , 49 );

echo('<td ' . $styl . ' width: 124px; height: 24px; color: white;
">'.$vysledok.'</td>');
                        }
                        echo('</tr>');
                }
                echo('</table>');

        echo date('d-m-y') ;
        echo "<br>";
        echo date('h:i:sa');
?>
</body>

</html>
```

Php server

Generátor čísel športky

8	3	6	45	48	46	44
38	42	48	24	42	34	7
24	41	38	44	49	48	48
10	30	47	32	27	21	1
30	44	1	43	11	34	24
37	16	46	21	18	13	4
47	45	44	5	31	7	26

10-02-17
04:31:21pm

DATABÁZY

Viď súbor mySQL podľa itnetwork