PHP Lietuviškai

Informacija iš interneto pakampių surinko, visa savaitgalį prie Easy PDF sėdėjo ir kankinosi Justinas L. aka scooox. Taigi, kad visi girdėtų, sakau: šitas dokumentas yra surinktas iš visų kampų ir aš nesisavinu sau jo autorystės ar kaip ten sakoma =). Visa info apie autorius galite rasti apačioje...

Taigi šiek tiek api šitą ebook'ą. Kažkada seniai kažkur internete radau toki archyvą, pavadinimu php_lt.zip. Pažiūrėjau - visai nieko, newbiams gerai. Taigi šitas php_lt.zip yra šio dokumento (knygos, knygelės, biblijos.. koks skirtumas) pagrindas. Taigi visą zip'ą sugrūdęs į pdf'ą nusprendžiau jį papildyti ir informacija iš geriausio lietuviško PHP resurso - beta.php.lt. Tai taip ir gavos šis failiukas, kaip sakant viskas viename... Taigi imkit mane ir skaitykit.. ir neuždavinėkit tada kvailų klausimų #php kanale :].

Turinys

1. Kas yra PHP?

PHP (PHP hypertext preprocessor) tai skriptinimo kalba (scripting language) kuri pačioje pradžioje buvo orentuota tik į iternetą, nors šiuo metu ją galima "drąsiai" pavadinti programavimo kalba, kadangi su ja pilnai galima programuoti ne tik internetui (web'ui). Pavyzdžiui su php-gtk galima kurti pilnavertiškas cross platformines programas su GUI (graphical user interface), bei consolines programėles mail wrapperius ir t.t. PHP skriptai yra interpretuojami ir įvykdomi serverio pusėje. Pvz.:

```
<html>
<head>
 <title>Pavyzdys</title>
</head>
<body>

  <?php echo "Sveikas aš esu PHP skriptas"; ?>

</body>
```

Po šio skripto įvykdimo rezultate (naršyklėje) turėsime:

  Sveikas aš esu PHP skriptas

Tai gana paprasta bet naudos iš to == nulis Pradžioje pagrindinis PHP privalumas prieš CGI buvo tai kad PHP skriptas galėjo būti lengvai įterpiamas į HTML kodą, kaip CGI reikėjo visą HTML outputint iš CGI skripto. Kuo toliau tuo rečiau HTML'as yra maišomas su HTML'u didesniuose projektuose, tam naudojami "Templait'ai" (šablonai?). Yra ir lietuviška templaitų klasė - phemplate. Apie ją daugiau informacijos rasite čia.

PHP skirtumas nuo JavaScript yra tai jog PHP skriptai yra atliekami serverio pusėje o vartotojui yra gražinamas rezultatas (HTML'u ar XML'u). Rezultatą vartotojas mato savo naršyklėje. Tuo tarpu kai Java Script yra pilnai perduodamas kliento mašinai (kompiuteriui) ir atliekamas klientinėje dalyje.

PHP sintaksės pagrindai yra paiimti iš: C, Perl, Java, todėl programavusiems šiomis kalbomis yra ypatingai lengva pradėti programuoti PHP.

1.1 Trumpa PHP istorija

Viskas prasidėjo 1994 metų rudenį, kai Rasmus Lerdorf'as nusprendė patobulinti savo Home-Page (asmeninį puslapį) ir parašyti asmeninį varikliuką kuris atlikinėtu paprasčiausius veiksmus. Šį varikliuką parengė 1995 metų pradžioje ir pavadino "Personal Home Page Tools". Šie "tools'ai" mokėjo ne itin daug, ir galimybės jo buvo tik pačios paprasčiausios, sudarė ją vos keletą makrosų. (Patį pirmąjį Rasmus Lerdorf'o pranešimą į www konferenciją apie savo "Personal Home Page Tools" 1995 metais. Galite paskaityti štai čia).

1995 metų viduryje atsirado antroji varikliuko versija kuri vadinosi PHP/FI version 2. FI - dėl vienos priežasties: tiesiog Rasmus'as padarė papildomą paketą kuris jau mokėjo interpretuoti formas (Form Interpretator). PHP/FI kompiliavosi į Apache Web serverį ir naudojo standartinį Apache API. PHP skriptai pasirodė spartesni serveryje nei CGI, nes nereikėjo kurti naujo process'o. Po truputi PHP pradėjo lygintis savo galimybėmis su Perl'u (populiariausia kalba CGI rašymui). Su laiku buvo pridėta ir daugelių duomenų bazių palaikymas (MySQL, Oracle ir t.t.), interfeisas GD bibliotekai (parašytas Rasmus'o ir iki šiol kuruojamas jo) leido dinamiškai kurti paveiksliukus. PHP/FI labai greitai pradėjo plisti.

1997 metų pabaigoje Zeev Suraski ir Andi Gutmans nusprendė perrašyti PHP varikliuką su tikslu ištaisyti daugelį klaidų bei pagreitinti PHP skriptų atlikimą serveryje. Po pusės metų, 1998..06 išėjo nauja versija kuri buvo pavadinta PHP3. NetCraft'o duomenimis PHP 1999 metais naudojosi jau 1 mljn. domenų.

1999 metų pabaigoje buvo vėlgi perrašytas PHP3 varikliukas (pavadinimu Zend Engine) ir išleista nauja versija PHP4. Skirtumai tarp šių versijų yra tikrai dideli, kartu su pagrindiniu PHP branduoliu atsirado daugybe extension'u ir jie vis dar nenusutojami tobulinti ir rašyti (Extension'u pvz: gd, xslt, sockets,imap,

daugelių DB support'ai ir t.t. ir t.t.).

Šiuo metu (2002 metų sausis) yra ruošiamas vėlgi patobulintas PHP varikliukas pavadinimu Zend Engine 2. Jis bus išleistas 2002 metų viduryje. Zend Engine 2 ir bus PHP5 "širdis bei smegenys". Pasak Zeev'o daugiausia darbo bus idėta į naujo objektinio modulio tobulinimą. Ištaisys Objekto kopijavimo atmintyje problema (dabar sunku atkasti ar tu sukuri jau esamo objekto kopiją), ideologija Zeev'as paėmė iš Java. Tiesiog kuriant objekta (new Object()) kintamasis nebus pats objektas o tiesiog referenc'as į jį, taip darant priskirimą naujam kintamąjam bus priskiriamas nebe pats objektas o jo referencas (objektas nekopijuojamas atmintyje). Taip pat atsiras destruktoriai, exception handler'iai (try/throw/catch) ir dauguma kitų pakeitimų (->_clone(),->_construct(), ->_destruct()...)

1.2 PHP privalumai ir trūkumai

Apie privalumus ir trūkumus prieš kitas panašaus tipo programavimo kalbas galite rasti internete. PHPEverywere turi puiku straipsnių rinkinį apie PHP bei cfm,perl,jsp,asp. Aš tiesiog paminėsiu pagrindinius privalumus ir trūkumus nelygindamas su kitomis kalbomis.

Privalumai:

a) php yra visiškai nemokamas;
b) php yra visiškai cross platforminis (veikia įvairiose operacinėse sistemose: Win, *nix, MacOS, Solaris, HP-UX, AIX ir t.t.);
c) php yra opensourc'inis projektas todėl jį developina didelė grupė žmonių, todėl iškilę bug'ai yra greitai ištaisomi, ir php sparčiai plečiasi;
d) kaip ir cross platforminis jis veiki ir ant daugelio WEB serverių: Apache, IIS, PWS, OmniHTTP, BadBlue ir t.t.;
e) išmokti PHP programavimo pagrindų yra labai lengva;
f) pasižymi dideliu greičiu serverio pusėje, bei dirbant su duomenų bazėmis;
g) nedideliuose projektuose PHP jūs paprastai galite įterpti į savo HTML'ą;
h) developer'iai prirašė extension'ų kiekvienam gyvenimo atvejui;
i) kadangi php programuotojų yra be galo daug, daugumą jau parašytų skriptų galite rasti internete: HotScripts, FreeScripts, PHPClasses.upperdesign.com ir t.t. ir t.t.

Trūkumai:

a) PHP == interpretatorius (nors tiesa sakant man tai netrukdo);
b) kadangi PHP yra opensourc'inis projektas, jūsų parašyti sourc'ai yra laisvai prieinami (skaitomi). Taip žinau yra ZendEncoder bet jis yra mokamas. Gerai kad nors jis yra;
c) deja php-gtk sourc'ai vis dar negali būti encodinam.**1. Įvadas**

2. Skriptai

Mes pabandysim paaiškinti "liaudiškai". Kas yra **PHP**, ir ką su šia programavimo kalba įmanoma padaryti? O gi galima padaryti beveik viską, ko jums gali prireikti internete: apsilankiusiųjų skaitliukai, svečių knygos, forumai, skelbimų lentos ir daug daug kitų programų. **PHP** skriptai turi išplėtimą .php Juos galima sutikti ir su kitokiais išplėtimais: .php .phtml .php3 .php4
Skripto kūnas gali būti įterptas į bet kurią html vietą. Kūnas (pats skripto turinys) yra tarp štai tokių simbolių:

```
<?php
// štai čia rašomas pats skriptas
?>
```

arba

```
<?php /*Čia rašomas scripto tekstas*/ ?>
```

arba

```
<? /*Čia rašomas scripto tekstas*/ ?>
```

arba

```
<script language="php">
// čia įterpiamas scriptas
</script>
```

arba

```
<%
// įterpiamas tekstas tarp ASP skliaustelių.
%>
```

Kaip ir kiekvienoje programavimo kalboje (bet ne visose), eilutės skiriamos kažkokiu skiriamuoju ženklu. **PHP** kalboje eilutės skiriamos kabliataškiu. Pvz.:

```
<?php
echo 'Sveikas Pasauli';
echo "TESTAS TESTAS";
echo "Labas rytas";
?>
```

Taip pat šioje programavimo kalboje yra ir komentarai (comments). Komentuoti savo skripto dalis galima štai taip:

```
<?php
echo "Tai yra testas"; // Komentaras sakinio pabaigoje
/* Komentaras per kelias eilutes,
jo pagalba galite pakomentuoti detaliau */
echo "Tai kitas testas";
echo "Paskutinis testas"; # Štai čia dar vienas komentaro tipas
?>
```

3. Kintamieji ir kintamųjų tipai

PHP palaiko štai tokius kintamųjų tipus, kurių jums turėtų visiškai pakakti kuriant jūsų norimą skriptą:

· **Integer**; // *sveikojo skaičiaus tipas*

· **Floating-point numbers**; // *slankaus kablelio skaičiai, arba kitaip - realieji skaičiai*

· **String**; // *eilutės tipas*

· **Array**; // *struktūrinio kintamojo tipas - masyvas*

· **Object**; // objekto tipas (objektiniam programavimui)

Dažniausiai kitamojo tipas yra nustatomas ne paties programuotojo, o skripto kompiliatoriaus. Jis pats priskiria kintamajam tinkama tipą. Jei jums reikia keisti kintamojo tipą programos metu, tai galite padaryti naudodamiesi **settype()** funkcija.

Integer (sveikojo skaičiaus tipas)

Šis tipas priskiriamas kintamąjam šiais atvejais:
$a = 1234; # *dešimtainis skaitmuo*
$a = -123; # *neigiamas skaitmuo*
$a = 0x12; # *šešioliktainis skaičiaus pavidalas (lygus 18 dešimtainėje sistemoje)*

Floating-point numbers (slankaus kablelio skaičiai, arba kitaip realieji skaičiai)

Štai du būdai, vaizduojantys realųjį skaičių.
$a = 1.234;

$a = 1.2e3;
String (eilutės tipas)

String tipas gali būti vaizduojamas programavimo kalbose dviem būdais:
1. tarp "viengubų" kabučių pvz.: **a$**='Testas';
2. tarp "dvigubų" kabučių pvz.: **a$**="Kitas testas";
Kaip **C** (**C++**) ir **Perl**'e, **PHP** naudoja štai tokius intarpus:
\n -nauja eilutė
\\ - back slash'as
\$ - dolerio ženklas (nes paprastai su juo į stringą įterpiamas kintamasis)
\" - kabutės (kad su jomis neužbaigtumėte savo stringo) *Šiuo ženklu taip pat galima pavaizduoti ir kitus norimus simbolius*.

Array (struktūrinio kintamojo tipas - masyvas)

Vienmatis masyvas
Masyvą galima įsivaizduoti kaip vieną ilgą elementų seką, kurie susiję savo kintamųjų tipais.
Pavyzdžiui, štai masyvas savaitės dienos;
["pirm", "antr", "trec", "ketvr", "penkt", "sest", "sekm"]

$savdienos[0]="pirm";
$savdienos[1]="antr";
$savdienos[2]="trec";
$savdienos[3]="ketvr";
$savdienos[4]="penkt";
$savdienos[5]="sest";
$savdienos[6]="sekm";

Štai taip įmanoma surašyti elementus į masyvą savdienos.
PASTABA: atkreipkite dėmesį į tai, kad PHP masyvas prasideda nuo indexo [0], o ne kaip pascal'yje ar kitur nuo [1].

Į vienmatį masyvą įrašyti elementus galima ir šiuo būdu:
$savdienos[ ]="pirm"; *// 0 elementas*
$savdienos[ ]="antr"; *// 1 elementas*
$savdienos[ ]="trec"; *// ir t.t.*
$savdienos[ ]="ketvr";
$savdienos[ ]="penkt";
$savdienos[ ]="sest";
$savdienos[ ]="sekm";

Masyvas gali būti rūšiuojamas šiomis funkcijomis: **asort()**, **arsort()**, **ksort()**, **rsort()**, **sort()**, **uasort()**, **usort()**, ir **uksort()**. Plačiau apie šias funkcijas skaitykite manual'e.
Taip pat per masyvo elementus įmanoma "vaiksčioti" štai šiomis funkcijomis: **next()** ir **prev()**

Dvimatis masyvas (matrica)

Dvimatis masyvas vaizduojamas panašiai:
$skaicius[1][2] = $kitas; *// dvimatis masyvas*
Daugiamačiai (trimačiai ir t.t.) masyvai vaizduojami analogiškai:
$masyvas**[1][2][3] = **$kitas; *// trimatis masyvas*

Objekto tipas
Sukurti objektą ir vėliau jį naudoti galima štai taip:

```
Class objektas
{
   function parasyti()
   {
      echo "SVEIKAS PASAULI";
```

```
   }
}
$naujas = & new objektas;
$naujas -> parasyti();
```

Programavimo kalbose yra du bjaurūs dalykai - pointeriai ir referencai. pirmasis tai lietuviškai verčiasi rodyklė, o antrojo lietuvinį pavadinimą užmišau :] todėl bus referencai. aukštesnio lygio kalbose pointeriai po truputį dinginėja, o jų vietą keičia referencai. php jie taipogi yra. šiaip tai viskas apie juos kaip ir apie visą kitką geriausiai paaiškinta manuale.
php operatorius = kuria duomenų kopiją, taipogi ir objektų, jei juos priskyrinėjam. php autoriai aiškina kad jiems taip gaunasi greitesnis kodas. bet objektų dažniausiai nereikia kopijuoti, o tik norime turėti kelis vardus rodančius į tą patį objektą. todėl jei nenorim kopijos, o tik nuorodos į objektą: $object = & $original_object;
netgi kuriant objektą su new: $obj = new Class(), $obj tėra kopija pirminio objekto gauto su new, todėl patartina prieš new dėti & : $obj = & new Class(); ypač kai konstruktoriui paduodamos kitos nuorodos.

PASTABA: kintamieji *$kint* ir *$Kint* yra visiškai skirtingi todėl nesusipainiokite juos naudodami.

4. Veiksmai su kintamaisiais

Kintamieji gali būti globalūs ir lokalūs. Pavyzdžiui:

```
$a = 1; /* globalus kintamasis */

function Testas ()
{
   echo $a; /* lokalaus kintamojo spausdinimas */
}
Testas ();
```

Kaip manote, ką atspausdins ši funkcija? O gi nieko, kadangi funkcijoje naudojamas lokalus kintamasis, o jis vis dar nėra apibrėžtas. Pakeiskime pavyzdį, kad jis veiktu:

```
$a = 1; /* globalus kintamasis */

function Testas ()
{
   global $a;
   echo $a;
}
Testas ();
```

Dabar funkcija atspausdins reikšmę 1.
Pirmame funkcijos variante yra sukuriamas statinis kintamasis, kuris funkcijai baigus darbą yra sunaikinamas. Statinį kintamąjį galima aprašyti ir taip:

```
function Testas ()
{
  Static $a = 0;
   echo $a;
  $a++;
}
```

Dabar vykdant funkciją yra sukuriamas statinis kintamasis, kuris yra atspausdinamas ir padidinama jo reikšmė vienetu, tačiau vos tik funkcija baigia savo darbą, šis kintamasis yra sunaikinamas.

Yra tokia sąvoka, kaip kintamojo kintamasis: Pvz.:

```
$sak = "Sveikas";
$$sak = "pasauli";
```

```
echo "$sak ${$sak}";
```

Ekrane pamatysime: Sveikas pasauli.
Pavyzdys su kintamųjų veiksmais:

```
function double($i)
{
   return $i*2;
}

$b = $a = 5; /* reikšmę penkis priskiria kintamiesiems $a ir $b */
$c = $a++; /* kintamajam $c priskirima reikšmė 6 (padidinta $a reikšmė) */
$f = double($c++); /* funkcija double gražina reikšmę du kartus didesnę (return) funkcijos reikšmė bus 6*2=12 */
$c += 10; /* $c reikšmė padidinama 10'čia */
$d=$a+$b; /* susumuoja */
```

Kitos operacijos su kintamaisiais:

```
$a + $b; //sumuoja
$a - $b; //atima
$a * $b; //daugina
$a / $b; //dalina
$a % $b; //randa modulį
```

Operacijos su eilutėmis (string):

```
$a = "Sveikas";
$b = . "pasauli"; // po šio sakinio kintamasis $b bus lygus "Sveikas pasauli"
```

Loginiai operatoriai:

```
$a and $b; //teisinga jei abu reiškiniai teisingi
$a or $b; //teisinga jei nors vienas iš reiškinių teisingas
$a xor $b; //teisinga kai kažkuris iš teiginių teisingas bet ne abu kartu
!$a; //neigimas, ne $a
$a&&$b; //AND
$a||$b; //OR
```

Lyginimo operatoriai:

```
$a == $b; //teisinga jei lygu
$a != $b; //teisinga jei nelygu
$a < $b; //teisinga jei $a mažiau $b
$a > $b; //teisinga jei $a daugiau už $b
$a <= $b; //teisinga jei $a mažiau arba lygu $b
$a >= $b; //teisinga jei $a daugiau arba lygu $b
```

5. Duomenų (kintamųjų) perdavimas iš formos skriptui

Padarykime paprastą mail'o siuntimo form'ą (skriptą). Mes naudosime paprastą **PHP** funkciją **mail()**. Ką ši funkcija daro…..Štai jos aprašymas:

```
mail($adress, $subject, $msg, "From: Nuo draugo.....");
```

Iš aprašymo turėtu būti viskas aišku :) Ši funkcija išsiunčia laišką adresatui. Mums lieka susikurt kintamuosius ir įvykdyti funkciją. Pradžioje sukursime skriptą mail.php3 , o vėliau ir formą, kurios pagalba bus perduodami duomenys:

```php
<?php
$address="jonas_jonaitis@centras.lt"; // čia adresas kur siunčiat
$subject="Laishkas nuo: $from"; /* kintamąjį $from skriptas ims iš formos */
mail ($address,$subject,$msg,"From: $from"); /* kintamasis $msg taip pat bus imamas iš formos */
?>
```

Štai ir sukūrėme skriptą. Jums lieka jį išsaugot vardu mail.php3. Dabar mums reikia sukurti formą, iš kurios bus siunčiamas laiškas. Čia ir sužinosite kaip kintamieji per formą perduodami skriptui (elementarus, lengvas dalykas) . Mums reikia sukurti formą su dviem laukais: nuo ko laiškas ir laiško turinys. Taip ir padarysime:

```
<FORM METHOD="POST" ACTION="mail.php3">
<I>- Įveskite duomenis:</I><BR>
<INPUT TYPE="text" size="20" NAME="from"><BR>
<TEXTAREA NAME="msg" ROWS="3" COLS="20"> </TEXTAREA> <BR>
<INPUT TYPE="Submit" NAME="Ok" VALUE="Įšsiūsti">
</FORM>
```

Štai ir viskas. Daugiau nieko daryti jums nebereikia. Juk taip paprasta? Kintamieji **from** ir **msg** automatiškai bus perduodami į skriptą, o funkcija **mail()** juos panaudos.
Kam naudotis kažkieno padarytais skriptais? Taigi visai paprasta norimą skriptą pasidaryti pačiam, pagal save. O tai atlikti galite tikrai labai nesunkiai ir per trumpą laiko tarpą.
Panašiai yra ir su adresinių duomenų perdavimu. Tarkime mums reikia, kad į administravimo skriptą ateitų duomenys **username** ir **pasword**. Tai galima padaryti kreipdamiesi į skriptą tokiu pavidalu: http://www.tarambaram.lt/darkasnors/admin.php3?user=jonas&pswd=jonaitis Tada skripte admin jūs gaunate kintamuosius **$user** ir **$pswd** ir galite juos sutikrinti su jūsų norimais.

5.1 Plačiau apie funkciją mail()

5.1.1 php.ini konfiguravimas

Visų pirma norėjau atkreipti visų dėmesį, kad dažniausiai jei jūs testuojates po Windows platforma jums mail() funkcija neveiks. Neveiks todėl kad php.ini faile pagal nutylėjimą SMTP serveris nurodytas localhost, o didžiausia tikymybė yra ta kad pas jus po localhost SMTP serverio nėra (nebent sėdite po Windows Server), todėl savo ekrane matysite tik štai tokį užrašą: ***Warning**: Failed to Connect in c:\kelias\kelias\failas.php on line XX*. Štai tai apie ką jums ir sakiau: php negali prisijungti prie SMTP serverio.
To galite išvengti įrašę php.ini faile SMTP=jums_žinomas_smtp_serverio_adresas. Jei pas jus PHP ne kaip CGI neužmirškite restartuoti Apache serverio.

5.1.2 Pradmenys

taigi, ką daro mail() funkcija? mail() funkcija leidžia jums siųsti el.pašta tiesiai iš jūsų kodo. Ji dažniausiai naudojama kontaktų, registravimo formose. Kai tam tikru meil'u norima informuoti apie užpildytą formą. Bet pritaikymo būdų yra begalė.

bool mail(string to, string subject, string message [, string additional_headers [, string additional_parameters])

mail funkcija gražina boolean tipą (true/false). Jei mail funkcija gražina false reiškia el. pašto išsiųsti nepavyko. Štai kaip siunčiamas paprasčiausias el.paštas:

```php
<?php
mail('webmaster@php.lt','Čia norima tema (subject)','Ir einanti žinutė');
?>
```

Taigi kaip matote pirmasis mail argumentas, tai el. pašto adresas kuriam siunčiate el.paštą, paskui eina Subject'as (norima tema), o po subject'o ir žinutės tekstas. Žinutės texte norint nukelti teksta į naują eilutę įterpkite \n (new line|nauja eilutė), pvz.:

```php
<?php $tekstas = "mano žinutė\n\nNikolajus Krauklis\naka DzHiBaS";?>
```

Po žinutės gali eiti papildomi parametrai: headeriai, bei kiti papilomi parametrui sendmail'ui ir t.t.

Tik išsiuntus šį el. paštą pastebėsite kad el.žinutėje nėra (kaikurių) lietuviškų raidžių. Jų nėra dėl netinkamos el.žinutės koduotės. Žinutės koduotė yra nustatoma žinutės headeryje (antraštėje).

5.1.3 Antraštės (headeriai)

pastaba: žinutės antraštės yra atskiriamos "\r\n".

headeriuose galima nurodyti daug įdomių bei naudingu dalyjų: nuo ko siunčiamas laiška, laiško koduotė, reply-to laukas, meileris, bcc ir t.t. Štai pavizdys paprasčiausio headerio kuris turėtu būti kiekviename laiške:

```
<?php
$header = "Content-type: text/plain; charset=\"windows-1257\"\r\n";
$header .= "From: Nikolajus Krauklis <mano@el_pastas.lt>\r\n";
$header .= "Reply-to: mano@el_pastas.lt\r\n";

mail('webmaster@php.lt','Cia norima tema (subject)','Ir einanti žinute',$header);

?>
```

Content type nurodome kad žinutė bus plain tekstu (ne HTML) bei windows-1257 charset'u. Išsiuntus laišką su šiuo headeriu esu isitikinęs kad windows vartotojai gaus jūsų išsiųsta el.žinutė su visom lietuviškom raidėm.
From headeryje nurodome nuo ko siunčiamas laiškas. Siuntėjas/gavėjas gali būti norodomas štai tokiais formatais:

```
<?php
/*
* paste iš kažkokio RFC susijusio su meil'ais
*
* From: mark@cbosgd.UUCP
* From: mark@cbosgd.UUCP (Mark Horton)
* From: Mark Horton <mark@cbosgd.UUCP>
* From: "Mark Horton" <mark@cbosgd.UUCP>
*/
?>
```

Reply-to tikriausiai jau patys supratote kam reikalingas? Pagal nutylėjima jei nėra reply-to jūsų meil'eris darys reply asmeniui kuris atsiuntė šį laišką, bei jei yra reply-to, paspaudus meileryje (outlook'e, bat'e, mutt'e ir t.t.) reply, tai to: lauke atsiras reply-to el.pašto adresas.
Štai dar papildomi headeriai kurie gali praversti:

```
<?php
$headers .= "X-Mailer: mano PHP meileris\r\n"; // maileris
$headers .= "X-Priority: 3\r\n"; // žinutės tipas: 1 UrgentMessage, 3 Normal
$headers .= "Return-Path: <mail@server.com>\r\n"; //kur gražinti meilą ištikus klaidai

mail('webmaster@php.lt','Cia norima tema (subject)','Ir einanti žinute',$headers);
?>
```

X-Mailer - tai niekur nedominuojantis headeris. Priority - dažnai pastebite kad būna žinutės su šauktuku šone, ten žymimi prioritetai. jei uždėsite prioritetą 1 - tai reiškia žinutė yra auksčiausio prioriteto ir prie el.žinutės gavėjo el.pašto programoje atsiras raudonas šauktukas. Dažniausiai visi siunčia el.žinutes su 3 (normaliu) prioritetu.
Return-path - tai el.pašto adresas į kurį bus nukreipiamos visos klaidas (pvz.: el.žinutė nepasiekė reikiamo nusmigusio serverio, arba tokio el.pašto adreso adresas jau nebeegzistuoja sistemoje)

Kaip nusisiųsti el.žinutės kopiją sau? Yra tris galimybės:
1. pats paprasčiausias tai nurodyti to lauką štai taip:

```
<?php
$to = "Kažkoks vartotojas <pvz@pvz.lt>" . ", ";
$to .= "Nikolajus Krauklis <webmaster@php.lt>";

mail($to,'Žinutė dviems gavėjams','Tekstas');
?>
```

2. savo el.pašto adresą nurodyti Cc headerio lauke:

```
<?php
$header .= "cc: webmaster@php.lt\r\n";
?>
```

Šiuo atveju visi gavėjai matys kam buvo siunčiamas gautas el.paštas.

3. Naudotis headerio Bcc lauku:

```
<?php
$header .= "bcc: webmaster@php.lt\r\n";
?>
```

dabar gavėjai niekaip nepastebės kad meil'o kopija buvo siųsta ir man!!! :)

5.1.4 Conclusion

na va berods viską ir išaiškinau. Norėjau pridurti ir tai kad el.žinutės gali būti siunčiamos ir HTML formatu, tada headeris atrodytu taip:
`<?php $header = "Content-type: text/plain; charset=\"windows-1257\"\r\n";?>`
Dabar el. žinutėje galite naudoti ir HTML tagus (pastaba: ne visi meileriai turi galimybė rodyti el.žinutes HTML formatu).

Gal pravers šie dokumentai: » RFC 1896, » RFC 2045, » RFC 2046, » RFC 2047, » RFC 2048, ir » RFC 2049. Jei jums reikia straipsnio kuriame aiškinama apie MIME el. pašto formatą bei jo headerius apsilankykite Zend'e.

6. Duomenų (kintamųjų) atėjusių iš formos patikrinimas

Dažnai tenka susidurti su tokia problema kaip vandalizmu internete. Žmonės bando apeiti formų pildymus, duoda "netikrus" el. pašto adresus ir t.t. Yra priemonių nuo to apsisaugoti.

Visų pirmą reiktu riboti žinutės ilgį. Kad nenaudėliai neprirašytų šiukšlių. Tai galima padaryti taip:

```
<INPUT TYPE="text" size="20" NAME="from" maxlength="30"><br>
<TEXTAREA NAME="msg" ROWS="3" COLS="20" maxlength="500"> </TEXTAREA>
```

maxlength - leidžia įvesti vartotojui tik limituotą simbolių skaičių.

E-mail'as be abejo gali būti nesąmoningas, todėl padarysime mažą skriptuką, kuris patikrina ar e-mail'as parašytas teisingai. Dabar išnagrinėsime patį paprasčiausią atvejį, kai vartotojas neįveda e-mail'o:
Tam mes naudosim funkcija **empty( )**; Ji tikrina ar kintamasis nėra tuščias (ar jam yra priskirta kokia nors reikšmė)

```
if (empty($from))
{
   echo "Jūs neįvedėte savo e-mailo";
}
```

Įterpkite šį gabaliuką į mail.php ir jūs jau turėsite apsaugą nuo tuščio **from** laukelio. Be abejo žmogus gali į laukelį įveti bet ką, tada bus apeita ši apsauga. Kad taip neatsitiktų, patikrinsime ar įrašytas e-mail'as turi savyje simbolį "@":
Naudosime paprastą string funkcija: **int strrpos**(**string** haystack, **char** needle); Ji randa ieškomo simbolio poziciją eilutėje:

```
function ar_yra_eta() /* Funkcija, kuri patikrina ar kintamajame $from yra "@" */
{
   global $from; /* $from padarome globaliuoju kintamuoju */
   $pos = strpos($from, "@"); /* į kintamąjį $pos įrašoma pozicija kurioje stovi "@" */
    if (!$pos) /* Pažiūrime ar apskritai yra "@" e-meile */
    {
      return false; /* Gražina funkcijos reikšmę false jei "@" nėra emeile */
    }
    else
   {
      if ($pos!=0 && $pos!=1)
      { /* Apsaugome, kad nebūtu pavyzdžiui toks meilas ( a@centras.lt arba   @centras.lt) */
          return true; /* Jei viskas tvarkoj gražina funkcijos reikšmę true. */
      }
   }
}
```

Štai padarėme funkciją, kuri apsaugo nuo e-mail'o be simbolio "@".

Taip pat galime tikrinti, ar į skriptą duomenys atsiunčiami POST metodu. Pvz. iš HTML dokumento:

```
<form method=post action=kazkas.php>
```

Pavyzdys iš kazkas.php skripto:

```
if ($REQUEST_METHOD == "POST")
{
   // kazka darom
}
else echo "Klaida! Metodas turi būti POST!";
```

Taip apsisaugosite nuo GET metodu perduodamų kintamųjų, pvz.
kazkas.php?vardas=siaubas&mailas=betkoks&kazkas=asdasd...

7. Darbas su data ir laiku

Darbui su data **PHP** turi savo funkciją **date**(formatas). Tai gana patogi funkcija. Funkcija naudojama laiko ir datos gavimui įvairiais formatais.
Formatai:

- Y - metai,pvz.: 1997
- y - metai,pvz.: 97
- M - mėnuo, pvz.: Oct
- m - mėnuo, pvz.: 10
- D - savaitės diena, pvz.: Fri
- l - savaitės diena, pvz.: Friday
- d - diena, pvz.:27
- z - metų diena, pvz.: 299

- H - valandos 24 val. formatu
- h - valandos 12 val. formatu
- i - minutės, pvz.: 5
- s - sekundės, pvz.: 40
- a - am/pm

programų pavyzdžiai:
$date = **date**('d M Y'); // *Į naršyklės ekraną išves: 22 Nov 1999*
$date = **date**('H:i:s'); // *Į naršyklės ekraną išves: 20:24:30*
$date = **date**('d M Y, H:i:s'); // *Į naršyklės ekraną išves: 22 Nov 1999, 20:29:42*
Bet dažnai mums reikia ne angliškos datos puslapyje, o lietuviškos. Kaip sulietuvinti datą? Skripto pavyzdys:

```
<?php
function data_lt()
{
   $menesis = date('n');
   $mas_menesiai = array("Sausio", "Vasario", "Kovo", "Balandžio", "Gegužės", "Birželio",
"Liepos","Rugpjūčio", "Rugsėjo","Spalio", "Lapkričio", "Gruodžio");
   $data = date('Y ');
   $data .= $mas_menesiai[$menesis-1];
   $data .= date(' j \d.');
   return $data;
}

echo data_lt();
?>
```

Skriptas išves į ekraną **2002 menesis diena d.**, pvz. **2002 Balandžio 7 d.**

8. Veiksmai su failais

Anksčiau (tačiau ir dabar dažnai sutinkamas) vienas iš pagrindinių būdų saugoti duomenis, buvo laikyti juos paprastame faile. Duomenys buvo skaitomi iš failo ir rašomi į failą, kol neatsirado greitesnis būdas - **DB** (duomenų bazės). Nors duomenų bazių vartojimas vis populiarėja, failų skaitymas ir rašymas į juos yra naudingi.
Pagrindinės funkcijos skaitymui bei rašymui: **fopen()**, **fgets()**, **fputs()**, **fclose()**, **feof()**; Jų pilnai pakanka dirbui su failais.

fopen()

fopen - atidaro failą (skaitymui, rašymui arba papildimui). **fopen**(string failo_vardas, string mode); mode - kokiai "sesijai" atidaryti failą. O "sesijos" yra štai tokios:
'**r**' - atidaro failą tik skaitymui, pointerį pastato failo pradžioje,
'**r+**' - atidaro failą skaitymui ir rašymui, pointerį pastato failo pradžioje,
'**w**' - atidaro failą rašymui, pointerį pastato failo pradžioje, failo dydį nunulina (ištrina visus buvusius duomenis), o jei failas neegzistuoja - sukuria jį,
'**w+**' - atidaro failą skaitymui ir rašymui, pointerį pastato failo pradžioje, failą nunulina, o jei failo nėra jį sukuria iš naujo.
'**a**' - atidaro failą tik rašymui, pastato pointerį į failo pabaigą, jei failas neegzistuoja - sukuria jį,
'**a+**' - atidaro failą ir skaitymui ir rašymui pastato pointerį į failo pabaigą, jei failas neegzistuoja - sukuria jį.

Pavyzdžiai:
$fp = **fopen**("/home/kazkas/file.txt", "r");
$fp = **fopen**("http://www.php.lt/", "r");

$fp = **fopen**("ftp://user:password@pvz.lt/", "w");

Atidarius failą, atlikus savo veiksmus, būtinai reikia jį uždaryti pasinaudojus funkcija **fclose()**;
Pvz.:
fclose($fp);

fputs(); - įrašo duomenis į failą,
fgets(); - nuskaito nurodyto ilgio eilutę į string tipo kintamąjį,
$string = **fgets**(**$fp**,255); // *nuskaito visą eilutę*

Paprastas pavyzdys:

```
$fp = fopen("/tmp/inputfile.txt", "r");
while ($bufferis = fgets($fp, 4096)) {
echo $bufferis;
}
fclose($fp);
```

Nuskaito visas failo eilutes ir išveda į ekraną. Bet ciklą galime pakeisti paprastesniu:

```
$fp = fopen("/tmp/inputfile.txt", "r");
while (!feof($fp)) {
$bufferis = fgets($fp, 4096)
echo $bufferis;
}
fclose($fp);
```

Dabar ciklas vykdomas tol, kol prieinama failo pabaiga.
Padarykime paprasčiausią puslapio skaitliuką, kad pamatytume kaip gi ištiesų veikia aukščiau aprašytos funkcijos:

```
<?php
$filename = "skaitlius.dat"; /* Priskiria kintamąjam failo pavadinimą */
$fp = @fopen($filename,"r"); // Atidaro failą skaitymui
if ($fp) { /* Jei failas atidarytas vykdomas sąlygos sakinys */
$counter=fgets($fp,10); /* Į kintamąji nuskaitoma skaitliuko reikšmė */
fclose($fp); // Uždaromas failas
} else {
$counter=0; /* Jei failo nebuvo skaitliui priskiriama nulis */
}
$counter++; /* Padidinama skaitliuko reikšmė vienetu */
echo $counter; // Išspausdinama reikšmė
$fp = fopen($filename,"w"); /* Ir toliau į failą įdedama nauja skaitliuko reikšmė padidinta vienetu. */
if ($fp) {
$counter=fputs($fp,$counter);
fclose($fp);
}
?>
```

Štai taip paprastai jūs galite pasidaryti savo puslapio skaitliuką. Man iškilo toks klausimas :/ O jeigu puslapis labai dažnai lankomas ir būtent tuo pačiu momentu ateina du lankytojai. Pirmasis lankytojas faile nuskaito (pvz.: skaitliuko reikšmę) o kitas įrašo. Kas tada? O gi tada kyla grėsmė prarasti skaitliuko reikšmę faile, kadangi negalima tuo pačiu metu ir skaityti ir rašyti į failą. Kad tai išpręstume, galime naudoti vieną funkciją **flock()**. Ši funkcija laikinai sustabdo antro vartotojo darbą ir palaukia kol pirmasis baigs darbą su failu, tada antrąjam vartotojui leidžiamas darbas su tuo failu. Patariama kuo dažniau vartoti šią funkciją dažnai lankomuose puslapiuose!!!
Pavyzdys:

```
$failas = @fopen('skaitlius.dat','r');
flock = ($naujas_failas,2);
$count = fgets($file, 255);
```

```
$count++;
flock = ($naujas_failas,3);
fclose($file);
```

Šiuo atveju jūs esate apsaugoti nuo galimo konflikto su failu.

Dauguma iš mūsų padėje mokintis PHP tikrai nepradėjome nuo PHP+SQL ar Oracle. Be to, gal ne kiekvienas turi serverį su duombazės palaikymu. Aš pats asmeniškai mažiausiai pusę metų "vargau" su tekstiniais failais. Kurį laiką man to tikrai pakako pakeisti duomenų bazę.

8.1. Informacijos perskaitymas iš tekstinio failo. Pritaikymas: counter'io rezultato bei ištisinio teksto nuskaitymas ir pan.

```
<?php
$failas="duomenys.txt";

$duomenys = fopen($failas, "r");
$informacija = fread($duomenys, filesize($failas));
fclose($duomenys);

echo $informacija;
?>
```

8.2. Naujas įrašymas ištrinant visus senus duomenis. Pritaikymas: paprasto counter'io skaičiaus įrašymas ir pan.

```
<?php
$informacija="Tekstas, irašomas i faila";
$failas="duomenys.txt";


$duomenys=fopen($failas, "w");
fwrite($duomenys, "$informacija");
fclose($duomenys);
?>
```

8.3. Įrašymas neištrinant senų duomenų, o tęsiant sąrašą toliau. Pritaikymas: įvairūs log'ai ir pan.

```
<?php
$informacija="Tekstas, irašomas i faila";
$failas="duomenys.txt";


$duomenys=fopen($failas, "a");
fwrite($duomenys, "$informacija\n");
fclose($duomenys);
?>
```

8.4. Tekstinė duomenų bazė. Informacijos eilutės perskaitymas iš tekstinio failo.
Pritaikymas: objekto aprašymas, jei objektas turi daugiau nei savybę

Pvz.: Mindaugas|20|185|70

Pateikta eilutė apibūdina asmens vardą, amžių, ūgį ir svorį.

Taigi, *duomenys.txt* struktūra turi būti tokia: *tekstas0|tekstas1|tekstas2|tekstas*

Kodo fragmento esmė tokia, kad yra perskaitoma eilutėje surašyta informacija ir išskirstoma į atskirus atitinkamus kintamuosius $laukelis[$i]. Šiuo būdu jau įmanoma sukurti panašią į duomenų bazės lentelę.

```
<?php
$failas="duomenys.txt";
$simboliu_skaicius = count($failas);
$stulpelis = file($failas);

for($i=0; $simboliu_skaicius > $i; $i++){
$laukelis = explode("|", $stulpelis[$i]);
}

echo "$laukelis[0] $laukelis[1] $laukelis[2] $laukelis[3]";
?>
```

Kaip įrašyti duomenis į failą, galite perskaityti **skyriuje 2.**
Tik kintamasis $informacija turėtų būti $informacija="Mindaugas|20|185|70";

8.5. Daug tekstinių eilučių. Pilnos duomenų bazės struktūra.

Pvz.:

```
Mindaugas|20|185|70
Kristina|20|174|50
Rimantas|50|190|80
[..]
Vaida|19|171|53
```

duomenys.txt failo sukūrimas yra paprastas:

```
<?php
$vardas="vardas";
$amzius="amzius";
$ugis="ugis";
$svoris="svoris";

$informacija=$vardas."|".$amzius."|".$ugis."|".$svoris;
$failas="duomenys.txt";


$duomenys=fopen($failas, "a");
fwrite($duomenys, "$informacija\n");
fclose($duomenys);
?>
```

9. REMOTE_HOST ir REMOTE_ADDR

Ar jums įdomu kas lanko jūsų puslapį? Manome, kad taip! Todėl skriptas, kuris atlieka loginimą buvo, yra ir bus visada akltualus. O tai nėra labai sunku. Pasinaudojus štai tokiomis nesunkiomis funkcijomis :

Getenv("**REMOTE_HOST**");
ir
Getenv("**REMOTE_ADDR**");

Mes galime sužinoti kompiuterio hostą ir IP adresą iš kurio buvo peržiūrinėtas puslapis.

```
$host = getenv('REMOTE_HOST');
$ip = getenv('REMOTE_ADDR');
```

Štai taip sužinome nutolusio kompiuterio duomenis. Lieka tik išsaugoti visa tai faile, kuris ir bus jūsų puslapio lankytojų logas. Failą dar papildysime data:

```
<?php
$filename= "log.dat";
$fp = @fopen($filename,"a+");

$host = gethostbyaddr($REMOTE_ADDR);
$ip=getenv('REMOTE_ADDR');
$date=date('d M Y, H:i:s');

$str = (" Data: $date \n IP adresas: $ip \n Host\'as: $host \n ------------------------------------
\n ");
fputs($fp,$str);
fclose($fp);
?>
```

Faile log.dat bus išsaugota kažkas panašaus į tai:

```
Data: 24 May 2000, 22:59:23
IP adresas: 000.000.000.000
Host: host.vu.lt
--------------------------------------
Data: 24 May 2000, 23:03:03
IP adresas: 000.000.000.000
Host: host.ktu.lt
--------------------------------------
Data: 24 May 2000, 23:04:16
IP adresas: 000.000.00.00
Host: host.ku.lt
--------------------------------------
```

10. Cookies

"Cookiai" yra dažnai vartojamas "daiktas". Tai mechanizmas, kuris saugo duomenis nutolusiame kompiuteryje. Tarkime, mums reikia žinoti kada žmogus paskutinį kartą buvo mūsų puslapyje. Tai įmanoma padaryti pasinaudojus "cookiais". Nustatyti "cookius" galima pasinaudojus **PHP** funkcija **SetCookie()**. "Cookiai" yra puslapio header'io ("galvos") dalis, todėl **SetCookie** funkcija turi būti panaudota prieš bet kokius duomenų siuntimus naršyklei (browseriui).
Pvz.:

```
SetCookie ("ManoCookis[ ]", "Testas", time()+3600);
```

Jei norite įterpti kelias reikšmes tiesiog pakartokite šia funkciją, bet jei "cookiu" vardai sutaps, bus pakeista tik reikšmė (replace) Jei norite naudoti savo pavadinimus pvz.: **pask_buvo**, tai įterpkite tai vietoj **ManoCookis**[ ].
Tarkime, mums reikia sukurti elektroninės parduotuvės vežimėlį. Jame bus saugoma informacija apie tai, kiek prekių vežamės ir visi jų pavadinimai:

```
$Kiek_prekiu++;
SetCookie ("Kiek", $Kiek_prekiu, time()+3600);
SetCookie ("Preke[$Kiek_prekiu]", $prekes_pav, time()+3600);
```

Štai taip atrodys skripto dalis.

"Cookių" gyvavimo laikotarpis nusistato taip (tarkime mums reikia, kad mūsų "cookiai" gyvuotų penkias dienas):

```
$k_gyvena = 5 * 24 * 3600; /* Gyvavimo laikotarpis nusistato sekundėmis; */
SetCookie ("Kiek", $Kiek_prekiu, time()+$k_gyvena);
```

11. Ciklai

Jei ne ciklai, programavimas būtų gana nuobodus užsiėmimas. Ciklas, tai priemonė, kurios pagalba galima vykdyti kodą nurodytą skaičių kartų arba tol, kol nebus įvykdyta kokia nors sąlyga. PHP kalboje yra du ciklų tipai: while ir for. Ciklai while tikrina sąlygas prieš arba po kiekvieno ciklo pakartojimo ir kartoja ciklą tik tuo atveju, jei sąlyga vis dar netenkinama. Cikle for, prieš pirmą ciklo pakartojimą, nurodomas ciklo kartojimų skaičius, kuris negali būti keičiamas ciklo vykdymo metu.

Ciklas while

Šio ciklo sintaksė panaši į operatoriaus if sintaksę.

```
while(sąlyga) {
...kodas...
}
```

Ciklo while sąlyga yra loginio tipo. Jeigu ji įgija reikšmę false, tai kodas parašytas rietiniuose skliaustuose praleidžiamas. Jei true - kodas riestiniuose skliaustuose vykdomas. Pasiekus ciklo uždarymo skliaustą }, reikšmė vėl tikrinama ir, jei ji teisinga ciklas kartojamas. Tokiu būdu ciklas yra kartojamas tol, kol [vykdoma sąlyga. Atkreipkite dėmsį į tai, jog sąlyga yra tikrinama kiekvieno ciklo pakartojimo pradžioje, todėl jei sąlyga bus pateinkinta jau kodo viduryje, tai kodas bus vis tiek vykdomas iki galo. Norint nutraukti ciklo darbą prieš laiką naudokite operatorių break.

```
$i=11;
while(--$i) {
if(mano_funkcija($i) == "error") {
break; //sustabdyti ciklą
}
++$kinamasis;
}
```

Jeigu numanoma mano_funkcija() funkcija, duotame pavyzdyje, neduos jokios klaidos, tai ciklas bus kartojamas dešimt kartų, tol, kol $i reikšmė bus lygi nuliui (priminimas: nulis grąžina reikšmę "false"). Jei funkcija mano_funkcija() grąžina klaidą, įvykdomas operatorius break ir ciklas sustabdomas. Tačiau kartais tereikia prašokti vieną ciklo kartojimą ir prieiti prie kito, nestabdant paties ciklo. Tam naudojamas operatorius continue.

```
$i=11;
while(--$i) {
if(mano_funkcija($i) == "error") {
continue; //pereiti prie kito ciklo kartojimo, nepadidinant $kintamasis reikšmės.
}
++$kinamasis;
}
```

Šis kodas taip pat yra vykdomas dešimt kartų, jei mano_funkcija() negrąžina klaidos. Tačiau priešingu atveju ciklas nėra stabdomas, o yra pereinama prie kito pakartojimo, nepakeičiant $kintamasis reikšmės.Jei po to $i reikšmė vis dar nelygi nuliui, ciklas kartojamas kaip įprasta.

Ciklas do...while

Ciklas do...while analogiškas while, tačiau sąlyga čia tikrinama ne pradžioje, bet pabaigoje kiekvieno pakartojimo. Tai reiškia, kad ciklas bus kartojamas bent vieną kartą:

```
echo("<SELECT name='dalys'>\n");
$i=0;
do {
echo ("\t<OPTION value=$i>$i</OPTION>\n");
} while (++$i < $kiekis);
echo("</SELECT>\n");
```

Vykdant šį kodą nulis bus išvedamas visada, kaip vienas iš pasirinkimų. Kitų pasirinkimų skaičius priklausys nuo $kiekis reikšmės.

Ciklai while ir do...while dažnai naudojami kartu su inkrementacijos(padidinimo vienetu) ir dekrementacijos(sumažinimo vienetu) operatoriais, siekiant nurodyti ciklo kartojimų skaičių. Operatoriai while taip pat dažnai naudojami skaitant duomenis iš duomenų bazės, eilutes iš failų, elementus iš masyvų(bet apie tai vėliau).

Ciklas for

Ciklo for sintaksė yra šiek tiek sudėtingesnė nei while, tačiau kartais naudoti for daug naudingiau.

```
for($i=1; $i<11; ++$i) {
echo("$i <br> \n"); // išvedami skaičiai nuo 1 iki 10
}
```

Ciklo for skliausteliuose nurodomi trys išsireiškimai, atskiriami kabliataškiais. Pirmuoju iš jų priskiriama reikšmė ciklo kintamajam. Šis išsireiškimas vykdomas tik prieš pirmą ciklo pakartojimą. Kitas išsireiškimas yra loginio tipo, ir yra tikrinamas kiekvieno ciklo pakartojimo pradžioje. Jei šio išsireiškimo reikšmė yra true, ciklas yra kartojamas, jei false, ciklo vykdymas sustabdomas. Trečiasis išsireiškimas vykdomas kiekvieno ciklo pakartojimo gale ir dažniausiai naudojamas ciklo kintamojo padidinimui arba sumažinimui.

Vidurinis išsireiškimas dažniausiai palygina ciklo kintamąjį su kokia nors anksčiau nustatyta reikšme, tačiau taip yr nevisada. Pavyzdyje pateiktas kiek kitokio panaudojimo pavyzdys.

```
for($i=1; mano_funkcija($i) != "klaida"; ++$i) {
// vykdomas kodas, kol mano_funkcija() negrąžina klaidos
}
```

Tą patį galima padaryti ir su ciklu while.

```
$i=1;
while (mano_funkcija($i) != "klaida") {
// vykdomas kodas, kol mano_funkcija() negrąžina klaidos
++$i;
}
```

Nėra tokios užduoties, kurią būtų galima atlikti su ciklu for, bet negalima atlikti su while. Tačiau dažniausiai ciklas for atrodo daug kompaktiškesnis ir labiau organizuotas.

Kaip ir kitose į C panašiose kalbose, PHP yra leidžiama (nors sutinkama retai) palikti vieną ar daugiau iš trijų minėtų išsireiškimų tuščiais.

```
for( ; ; ) {
if(mano_funkcija() == "stop") break;
}
```

Jeigu loginis išsireiškimas yra tuščias, tai pagal nutylėjimą jis grąžina true. Tai gali sąlygoti ir begalinio ciklo atsiradimą, jei nenaudosime break, return arba exit. Be abejo, pagrindas tam, kad nenaudoti kurio nors iš išsireiškimų cikle for, atsiranda retai.

12. Sąlygos operatoriai

Šioje pamokoje aptarsime PHP kalboje egzistuojančius operatorius, kurie sudaro programos griaučius. Sąlygos operatoriai ir ciklai padeda išpręsti daug užduočių.

Sąlygos operatoriai

Sąlygos operatoriai leidžia rašyti kodą, kuris bus vykdomas tik esant kažkokioms sąlygoms. PHP kalboje yra dvi sąlyginės konstrukcijos. Pirmoji iš jų yra if...elseif...else, leidžianti patikrinti keletą išsireiškimų ir vykdyti kodą, priklausomai nuo tų išsireiškimų reikšmės. Jeigu reikia palyginti vieną išsireiškimą su keletu reikšmių,

tai PHP leidžia naudotis konstrukcija switch....case, atliakančia šią užduotį.

Operatorius if

Operatorius if yra viena iš svarbiausių galimybių visose programavimo kalbose.Jis leidžia vykdyti tam tikras kodo eilutes, tik esant nurodytoms sąlygoms. Operatoriaus if sintaksė:

```
if (sąlyga) {
....kodas...
}
```

Jeigu sąlygos sakinyje vykdome daugiau nei vieną kodo eilutę, kodą reikia apskliausti riestiniais skliaustais. Jei vykdome tik vieną eilutę, to daryti nėra būtina.

```
// šis kodas išves "Lietuva.", jei kinamojo $salis reikšmė bus "lt".
if ($salis == "lt")
echo("Lietuva.");

// šis kodas išves "Lietuva. Jos sostinė - Vilnius", jei kinamojo $salis reikšmė bus "lt".
if ($salis == "lt") {
echo("Lietuva. ");
echo("Jos sostinė - Vilnius");
}
```

Operatoriuje if tikrinama sąlyga turi turėti loginio tipo(boolean) reikšmę true arba false. Bet kokia neįvykdoma sąlyga, nulis, tuščia eilutė, nenustatytas dydis ir PHP konstanta false turi reikšmę false. Pavyzdžiui, visi žemiau pateikti išsireiškimai turės reikšmę false:

```
if (5<4) echo ("Tai nebus išvesta");
if (false) echo ("Tai nebus išvesta"); //false yra PHP konstanta
if ("0") echo ("Tai nebus išvesta"); //Apdorojant eilutes, "0" virsta 0
if ($g) echo("Tai nebus išvesta"); //Tuo atveju jei $g nėra priskirta reikšmė
```

Dydis true yra ekvivalentus bet kokiai reikšmei, nelygiai nuliui, tuščiai eilutei, svarbu kad sąlyga bšti patenkinta. Žemiau pateikti išsireiškimai įgyja reikšmes true:

```
if ("false") echo ("Tai bus išvesta); // false jau yra nebe konstanta, o eilutė
if ("00") echo ("Tai bus išvesta); // eilutė iš dviejų nulių nėra perdirbama į sveikojo tipo reikšmę
if (0 ==0) echo ("Tai bus išvesta); // nulis lygus nuliui, todėl sąlyga patenkinta
```

Tačiau sąlygas galima padaryti ir sudėtingesnes, apjungiant kelias sąlygas loginiais operatoriais.

```
if (((4<5) && (3>2)) xor (5 == 5)) echo ("Tai išvesta nebus"); // teisingos abi sąlygos, apjungtos operatoriusmi xor, todėl viso išsireiškimas neteisingas
```

Paneigtos sąlygos

Jeigu tikrinama sąlyga grąžina false, tai PHP leidžia nurodyti kitą kodo bloką, kuris bus vykdomas, panaudojant else sakinį. Pavyzdžiui:

```
if ($k < 0) {
echo ("Neigiamas skaičius);
} else {
echo ("Teigiamas skaičius);
}
```

Duotasis sakinys tikrina, ar kintamojo reikšmė mažesnė už nulį. Jei taip, tai išvedama "Neigiamas sakičius". Jei yra priešingai, išvedama "Teigiamas skaičius". Egzistuoja sakinys elseif, leidžiantis patikrinti alternatyvias sąlygas. Pvz.:

```
if ($k < 0) {
echo ("Neigiamas skaičius);
```

```
} elseif ($k ==0) {
echo (”Nulis”);
} else {
echo (”Teigiamas skaičius);
}
```

Pavyzdyje pateikta konstrukcija tikrina, ar kintamojo reikšmė mažesnė už nulį. Jei taip, parašoma, jog tai neigiamas kaičius. Jei nepatenkinama pirmoji sąlyga, tikrinama, ar kintamais lygus nuliui. Jei taip, parašoma jog tai nulis. O jeigu nepatenkinama nei viena iš ankstesnių sąlygų, parašoma, jog tai teigiamas skaičius.

Operatorius switch

Tarikime, jog turime kintamajį $salis, kuriame yra saugomas sutrumpintas valsybės pavadinimas, tačiau mes norime išvesti pilną tos šalies pavadinimą. Jei darbą atliktume su konstrukcija if...elseif....else, tai atrodytų maždaug taip:

```
if ($salis== “ca”) {
echo (”Kanada”);
} elseif ($salis == “cr”) {
echo (”Kosta Rika”);
} elseif ($salis == “de”) {
echo (”Vokietija”);
} elseif ($salis == “uk”) {
echo (”Didžioji Britanija”);
} else {
echo(”JAV”);
}
```

Šiame pavyzdyje, mes kiekvieną kartą lyginome kintamajį $salis su tam tikra reikšme ir priklausmai nuo to, išvesdavome reikšmę. Tai yra pakankamai neefektyvu. Tai galima ištaisyti, naudojant konstrukciją switch/case. Operatorius switch naudojamas tada, kai reikia vieną kintamajį palyginti su keliomis reikšmėmis.

```
switch($salis) {
case “ca” :
echo (”Kanada”);
break;
case “cr” :
echo (”Kosta Rika”);
break;
case “de” :
echo (”Vokietija”);
break;
case “uk”:
echo (”Didžioji Brotanija”);
break;
default:
echo (”JAV”);
}
```

Operatorius switch ima reikšmę iš kintamojo $salis ir lygina ją su reikšmėmis pateiktomis case sakiniuose. Kai randama sutampanti reikšmė, vykdomas kodas, kol randamas operatorius break. Jei nėra sutampančių reikšmių vykdomas default sakinyje parašytas kodas.

Na, o dabar smulkiau panagrinėsime operatorių break. Jis leidžia sustabdyti operacijas case operatoriuose. Jei $salis reikšmė yra cr, tai išvedama "Kosta Rika", ir operatorius break sustabdo tolesnį vykdymą. Jei break neegzistuotų, programa būtų vykdoma toliau ir į ekraną būtų išvesti visų likusių valstybių pavadinimai. Tai gali būti naudinga, bet gali ir pakenkti. Mūsų atveju, tai pakenkė. Tačiau šią problema gali mums padėti išspręsti kitą užduotį:. mes galime apjungti kelis operatorius case ir jei bent vieno iš jū reikšmė atitiks reikiamą, bus vykdomas kodas.

```
switch($salis) {
case “ca” :
case “cr” :
case “us” :
echo (”Šiaurės Amerika”);
break;
case “de” :
case “uk” :
echo (”Europa”);
break;
}
```

Jei kintamojo $salis reikšmė yra ca, cr arba us, tai į ekraną išvedama “Šiaurės Amerika”, jeigu de arba uk, išvedama “Europa”.

13. Sesijos

Šis straipsnis yra parašytas remiantis tik teorinėmis žiniomis apie sesijas, neturint jokios patirties. Taigi nepykite, jei kai kas bus netikslu ar klaidinga. Būsiu dėkingas visiems išmanatiems sesijas už pastabas.
Na, o dabar prie reikalo. Su PHP4 atsirado nauja galimybė - integruotas sesijų valdymas, kas leidžia saugoti kintamuosius sesijos "objekte" viso apsilankymo metu.
Sesija - tai procesas, prasidedantis, kai lankytojas ateina į tinklapį, ir pasibaigiantis, kai tinklapis paliekamas (arba koks nors puslapis nutraukia ją). Dažniausiai cookie 's yra "pririšamas" prie naršyklės, o serveryje išskiriama vieta saugoti sesijos kintamiesiems. PHP4 naudoja failus, tačiau teoriškai galima naudoti ir duomenų bazes ir atmintinę, sesijos kintamiesiems saugoti.
Visi puslapiai naudojantys sesijas turi iškviesti funkciją session_start(), kuri pasako PHP varikliui, kad reikia pakrauti sesijos informaciją į atmintį.

Sesijos kintamieji

Sesijos kintamieji - tai globaliniai kintamieji, kurie saugo kintamųjų reikšmes per visus tinklapius, naudojančius sesijas. Šie kintamieji užregistruojami, naudojant funkciją session_register("kintamasis"). Nuo šiol visuose puslapiuose, kuriuose yra sesijos (su session_start()), bus galima naudoti kintamąjį $kintamasis su ta reikšme, kuri buvo priskirta jam prieš užregistravimą. Pvz.:

```
<?
session_start();
$kintamasis="Kintamojo reikšmė";
session_register("kintamasis");
?>
```

Kaip matome iš pavyzdžio, registruojant sesijos kintamąjį, jo vardas yra nurodomas be $ ženklo. Dabar mes panaudosime $kintamasis kitame puslapyje:

```
<?
session_start();
echo "Užregistruoto kintamojo reikšmė $kintamasis";
?>
```

Kai kintamasis tampa nebereikalingas, jį galima sunaikinti su funkcija session_unregister("kintamasis"). Taip atlaisvinama dalis atminties, kuri susieta su kintamuoju.

Sesijos pabaiga

Sesijos pabaiga nėra automatizuota. Serveriui sunku numanyti, kada vartotojas užbaigia sesiją. Yra keletas komandų padėsiančių pabaigti sesiją:

1. Funkcija session_destroy(). Ją iškvietus sesija baigiama.
2. Jei sesijos ID yra saugomi cookie' iuose, galima pakeisti jų gyvenimo laiką, taip įtakojant sesijos veikimo

laiką.
3. Galima koreguoti php.ini failą. gc_maxlifetime reikšmė nurodo, kiek laiko po paskutinio naudojimosi sesijos duomenimis jie yra ištrinami. Toks šiukšlių išvalymas reikalingas todėl, kad serveris nežino ar cookie' s vis dar yra kliento pusėje. Jeigu norima ištrinti senus sesijos duomenis, kai yra gaunami nauji, gc_probability savybei priskiriama reikšmė nuo 1 iki 100. Jei reikšmė 100, tai seni duomenys pakeičiami naujais po kiekvienos naujos užklausos. Jei reikšmė 1 (pagal nutylėjimą), senų sesijų duomenys bus ištrinami tik vieną kartą per 100 užklausų.

Jeigu sesijos ID saugoti, vietoj cookie'ų naudojami GET arba POST užklausos, reikėtų atkreipti ypatingą dėmesį į šiukšlių išvalymą, nes lankytojas gali pasižymėti puslapį su senu sesijos ID, ir jei nebus išvalyti seni sesijos duomenys, lankytojas tes seną sesiją, kas gali būti nenaudinga jums.

Sesijų ID

Sesijų ID gali būti saugomi trimis būdais:

1. Cookie' iais (pagal nutylėjimą)
2. Per GET/POST metodus
3. Automatiškai arba rankiniu būdu perrašinėjant URL.

Kai sesijos ID sagomi cookie'iuose, jums nereikia rūpintis niekuo, nei konfigūracija, nei papildomo kodo rašymu. Kitas dažnai naudojamas būdas yra rašyti ID GET užklausoje. Tada URL atrodys maždaug taip: skriptas.php?=. URL galima perrašinėti naudojant globalinę konstantą SID:

```
<?
printf("<a href=skriptas.php?%s>"Nuorodoa</a>, SID);
?>
```

arba funkciją session_id():

```
<?
$id=session_id();
print "<a href=skriptas.php?$id>Nuoroda</a>";
?>
```

Sesijų teorijos tiek. php.lt yra straipsnis apie prisijungimą su sesijomis.

Straipsniai apie sesijas:

http://www.phpdeveloper.org/view_tut.php?id=31
http://www.zend.com/zend/tut/session.php
http://www.phpbuilder.com/columns/mattias20000105.php3?print_mode=1
http://www.phpbuilder.com/columns/mattias20000312.php3?print_mode=1

14. Objektinis programavimas

Dabar siūlau susipažinti su pagrindiniais objectinio programavimo terminais:
Klasė - tai rinkinys funkcijų ir kintamųjų, bendradarbiaujančių tarpusavyje.
Savybės - tai klasėje naudojami kintamieji.
Metodai - tai klasėje naudojamos funkcijos.
Konstruktorius - tai funkcija įvykdoma sukuriant klasę (sukuriant objektą).
Iš šių painių apibrėžimų sunku ką nors suprasti. Bet manau, jog perskaitę šį straipsnį susidarysite bent minimalų vaizdą apie objektinį programavimą (OOP - Object Oriented Programming).
Taigi, pradėkime nuo sintaksės. Klasės aprašomos naudojant operatorių Class. Klasių kodas rašomas tarp riestinių skliaustų, lygiai kaip ir aprašant paprastas funkcijas:

```
<?
class klase{
// čia aprašomos objekto savybės
```

```
var $savybe1;
var $savybe2;

// čia aprašomi metodai
function metodas($kint)
{
// kodas
}
}
?>
```

Kaip matome, savybės yra aprašomos naudojant operatorių var. Dabar pateiksiu pavyzdį iš realaus gyvenimo, kurį būtų galima susieti su OOP. Automobilis. Tai bus mūsų objektas. Kaip žinome jis turi keletą savybių: spalvą, formą, dydį, galią ir t.t. Tai bus mūsų objekto savybės. Taip pat jis gali atlikti keletą veiksmų: pajudėti iš vietos, pasukti, sustoti, važiuoti. Visa tai bus objekto metodai. Dabar visa tai aprašykime PHP kodu:

```
<?
class automobilis
{
// automobilio savybes
var $pavadinimas;
var $spalva;
var $forma;
var $dydis;
var $galia;

function pajudeti()
{
// kodas
}

function sustoti()
{
// kodas
}

function vaziuoti($greitis)
{
//kodas
}

function pasukti($kampas)
{
// kodas
}
}

?>
```

Kaip matome, viskas pakankamai paprasta. Taigi dabar panaudosime šį objektą. Sukurkime keletą automobilių:

```
<?
// sukuriame mano automobilį
$mano=new automobilis;
$mano->pavadinimas="BMW";
$mano->spalva="Juoda";
$mano->galia="200 AG";
```

```
// sukuriame jos automobilį
$jos=new automobilis;
$jos->pavadinimas="Ferrari";
$jos->spalva="Raudona";
$jos->galia="300 AG";

// na, o dabar mamo automobilis turėtų pajudėti iš vietos
$mano->pajudeti();
// o dabar padidinti greitį iki 150 km/h
$mano->vaziuoti(150);

// o, štai atlekia jos automobilis
$jos->vaziuoti(180);

// o ne, keliu policija.....
$mano->sustoti();

?>
```

Naujas objektas sukuriamas panaudojant operatorių new. Kintamajam priskiriama "object" tipo reikšmė, taip paverčiant jį objektu. Panaudojant -> (priešingai ne JavaScript ir kt.) mes galime pasiekti visas objekto savybes ir metodus. Reikšmių priskyrimas objekto savybėms yra identiškas reikšmių priskyrimui kintamiesiems.
Naudojant OOP dažnai reikia keisti objekto savybių reikšmes pačios klasės viduje. Tam yra naudojamas išskirtas kintamasis $this, nurodantis, jog savybė priklauso esamai klasei.

```
<?
class automobilis
{
var $pavadinimas;
var $spalva;
// ..... kitos savybes .....

function perdazyti($spalva)
{
$this->spalva=$spalva;
}
}

$auto=new automobilis;
$auto->spalva="Raudona";
$auto->perdazyti("Juoda");
?>
```

Panaudojant $this galima ne tik naudoti objekto "vietines" savybes. Taip pat galima vykdyti ir metodus.
Na, o dabar pavyzdys iš praktinio klasių pritaikymo: sukursime klasę, kuri sukurs HTML lentelę, su nurodytu skaičiumi eilučių ir stulpelių.

```
<?
class lentele
{

var $eil;
var $stulp;

// nurodyti eilučių ir stulpeių skaičių
function nustatyti($eil, $stulp)
{
```

```
$this->eil=$eil;
$this->stulp=$stulp;
}

// sukurti lentelę
function kurti_lentele()
{
echo "<table border=1>";

for ($x=1; $x<=$this->eil;$x++)
{
echo "<tr>";
for ($y=1; $y<=$this->stulp;$y++)
{
echo "$x, $y";
}
echo "</tr>";
}
echo "</table>";

}

}
?>
```

Šioje klasėje yra du metodai: nustatyti(), kuris nurodo, kiek eilučių ir stulpelių turės lentelė, ir kurti_lentele(), kuris atvaizduoja lentelę pagal nurodytus duomenis. Štai panaudojimas:

```
<?
$obj=new lentele;
$obj->nustatyti(2,3);
$obj->kurti_lentele();
?>
```

OOP taip pat galima aprašytiyti funkciją, kuri bus vykdoma sukuriant objektą. Tokia funkcija vadinama konstruktoriumi ir turi turėti tokį pat pavadinimą, kaip ir klasė.
Pavyzdžiui, papildysime lentelės klasę funkcija, kuri priskirs reikšmes kintamiesiems pagal nutylėjimą.

```
<?
class lentele
{
var $eil;
var $stulp;

// dabar reikia konstruktoriaus
function lentele()
{
$this->eil=5;
$this->stulp=3;
}
}
//kitos funkcijos
?>
```

O dabar galime kurti lentelę, nenurodinėdami parametrų:

```
<?
$lent=new lentele;
$lent->kurti_lentele();
```

```
?>
```

Įvykdę šį kodą gausime 5x3 lentelę.

Na, ir galiausiai papasakosiu apie vieną iš svarbiausių OOP galimybių - plečiamumą ir paveldimumą. Tai yra labai paprasta: paprasčiausiai sukuriame klasę, kuri išplečia jau esamą ir paveldi jos metodus, savybes. Išplėsti jau esamai klasei yra naudojamas operatorius "extends". Taigi dabar išplėsime lentelės klasę:

```
<?
class lentele
{

var $eil;
var $stulp;
var $krastine;

// nurodyti eilučių ir stulpeių skaičių
function nustatyti($eil, $stulp)
{
$this->eil=$eil;
$this->stulp=$stulp;
}

// sukurti lentelę
function kurti_lentele()
{
echo "<table border=$krastine>";

for ($x=1; $x<=$this->eil;$x++)
{
echo "<tr>";
for ($y=1; $y<=$this->stulp;$y++)
{
echo "$x, $y";
}
echo "</tr>";
}
echo "</table>";

}

}

class geresne_lentele extends lentele
{
function nurodyti_krastine($storis)
{
$this->krastine=$storis;
}
}

?>
```

Tai tiek apie OOP. Kaip matote tai nėra itin sudėtinga, tačiau reikia nuolat tobulintis, praktikuotis, dirbti....

15. Tips and Tricks

Čia rasite patarimų, skriptų ir t.t.

15.1 parent::

Šitas tik ant senų php versijų rodos pradedant 4.0.4 arba net 4.0.5. kai turi kažkokią klasių hierarchiją ir iš paveldėtos klasės nori iššaukti tėvinį metodą, teoriškai turėtum rašyti parent::metodas();, bet, deja, jei tėvinė klasė yra ne tam pačiam kataloge, gausi klaidą, kad neįmanoma rasti tėvinės klasės, todėl reikia kviesti naudojantis konkretų tėvinės klasės vardą: tėvinėKlasė::metodas();.

15.2 Pranešimai apie klaidas

apie klaidas tai turiu tik vieną, bet super patarimą: visada užsidėk error_reporting (E_ALL). iš pradžių bus bjauru, nes rašant kodą belekaip, php mes dafiga klaidų, bet reikia tik įprasti atitinkamai programuoti.

tipiniai pataisymai:

- negalima naudoti kintamojo, prieš tai jo neapibrėžus. todėl visiems ateinantiems iš formos kintamiesiems reikia priskirti default reikšmes, nes jei vartotojas nieko neįvedė, tai kintamasis nebus net užsetintas. if (empty($var)) $var = '';
- $masyvas[indeksas] -> $masyvas['indeksas'], nors pirmas variantas ir veikia, bet php pirma ieško konstantos indeksas, neradęs praneša klaidą, ir verčia indeksą į string.
- isset($var) - patikrinti ar toks kintamasis išvis egzistuoja.
- @funkcija(), jei neišeina išvengti klaidų pranešimų vykdant funkciją, operatorius @ panaikina juos. tik nereikia persistengi, nes gali pakliūti į tokią padėtį, kai tavo skriptas sustoja neišmetęs jokio pranešimo.
- if ($kazkas) -> if (!empty($kazkas)), funkcija empty() negrąžina klaidos, jei kintamasis neegzistuoja.
- if (!$kazkas) -> if (empty($kazkas))
- for($i; $row = $list[$i]; $i++) -> for($i; isset($list[$i]); $i++) { $row = $list[$i]; }, bandant įvykdtyti sąlyga $row = $list[$i] po paskutinio elemento, gausit pranešimą, kad toks indeksas neegzistuoja, todėl priskyrimą reikia vykdyti ciklo viduje, o sąlygoje tik tikrinti ar egzistuoja toks indeksas.

15.3 Search Engine Friendly PHP pages

Įvadas

Kai rašote skriptus svarbus dalykas yra galimybė perduoti kintamuosius iš vieno skripto į kitą. Vienas iš metodų kaip tai padaryti - GET metodas. Būtent apie jį čia ir papasakosiu. Kai kurios paieškos sistemos mėgsta neindeksuoti puslapių kurie yra su GET metodų parametrais. Jei jūs neisitikinę kaip tiksliai atrodo GET metodu perduodami kintamieji, štai jums pavyzdys:

http://beta.php.lt/skriptas.php?page=news&id=2

Čia skriptui (skriptas.php) yra perduodami du kintamieji: $page bei $id. Juos automatiškai turėsite savo skripte: `$page == 'news' && $id == 2` .

Alternatyva įprastam GET metodui

Tai kaip gi galima apeiti štai rokį url'ą ? Ir kaip tada mes perduosim parametrus kitam skriptui ? Štai jums pavizdys kaip atrodo linkas kurį paieškos sistema jau supras kaip normalų:

įprastu GET metodu:

http://beta.php.lt/skriptas.php?page=news&id=2

mūsų variantai:

http://beta.php.lt/skriptas.php/page/news/id/2
http://beta.php.lt/skriptas.php/page.news/id.2
http://beta.php.lt/skriptas.php/page-news;id-2

arba

http://beta.php.lt/skriptas/page,news;id,2 (beta.php.lt variantas)

Čia jau kaip sugalvosite. Dažniausiai visi naudoją patį pirmąjį variantą, bet galima naudoti koki tik sugalvosite.

Kodėl vienas variantas liko be .php išplėtimo? Tai įmanoma padaryti Apache web serveryje, kai ant direktorijos Options parametruose yra nustatyta MultiViews arba Options All. (apache httpd.conf http://httpd.apache.org/docs/content-negotiation.html). Tada jei serveris neranda skriptas direktorijos, bando ieškoti failo skriptas.* Mūsų atveju jis randa skriptas.php

Kaipgi gauti visus parametrus po skriptas.php? Galima pasinaudoti štai tokiais PHP kintamaisiai: $PATH_INFO, $REQUEST_URI, $SCRIPT_NAME. Pažiūrėkime kaipgi veikia jie ir kaip mes visa tai realizuosime:

```
<?php
echo $PATH_INFO."<br>";
echo $REQUEST_URI."<br>";
echo $SCRIPT_NAME;
?>
```

Šį skriptą savo web serveryje pavadiname url.php ir žiūrime rezultatus. Įvedame paprasčiausią variantą: http://localhost/url.php/pirmas/pirmo_reiksme/antras/antro_reiksme (kurie bando ant savo darbinių kompiuterių galite pasitestuoti ir MultiViews tada jūs galite kreiptis į skriptą http://localhost/url/pirmas/pirmo_reiksme/antras/antro_reiksme) Rezultate browseryje matysime kad visi kintamieji mums gražina tai ko mums ir reikia. Patogesnis variantas yra naudoti $PATH_INFO, kadangi prieš tvarkantis su kintamaisiai nereikės iš gauto url'o striptinti(pašalinti) skripto vardo, mūsų atveju -> url.php

Dabar reikia padaryti taip kad mes galėtume naudotis ir pačiais kintamaisiai? Tai reiškia kad mūsų skriptuose atsirastų kintamieji $pirmas ir $antras ir jų reikšmės būtų analogiškos: $pirmas == 'pirmo_reiksme' && $antras == 'antro_reiksme';

Tam galime padaryti štai tokią funkciją:

```
<?php
function decode_url() {
global $PATH_INFO;

// Jei PATH_INFO - yra. Jei yra uzhsetinti parametrai

if(isset($PATH_INFO)) {

// Skaidom URL'a musu atveju pagal '/'
$duomenys = explode('/', $PATH_INFO);

// Skaichiujam kiek paramtru ir ju reikshmiu atejo ish url'o
$num_param = count($duomenys);

if($num_param % 2 == 0) {
$duomenys[] = '';
$num_param++;
}

// Padarom kad atsirastu kintamieji $pirmas bei $antras bei ju reiksmes
```

```
for( $i = 1; $i < $num_param; $i += 2) {
$GLOBALS[$duomenys[$i]] = $duomenys[$i+1];
}
}

}
?>
```

Štai šią funkcija įdedam į kokį nors header failą, bei includiname kiekviename skripte. Iš kart po jos decodinam url'a parašę: decode_url();

Štai variantas kai turime tik vieną skirybos ženklą '/', o kaip padaryti su keliais? Pavyzdžiui url.php/pirmas,pirmo_reiksme;antras,antro_reiksme ? Šiuo atveju $PATH_INFO == '/pirmas,pirmo_reiksme;antras,antro_reiksme';

Analogiškai darome funkciją:

```
<?
function decode_url()
{
// darom kad PATH_INFO prieinamas butu ir funkcijoje
global $PATH_INFO;

$duomenys = substr ($PATH_INFO , 1);

// skaidom url'a pagal pirma skirybos zhenkla ';'
$duomenys = explode(";",$duomenys);

// jei duomenu yra vikdysim toliau
if (isset($duomenys))
{
// zhiurim kiek yr tu variablu
$d_size = sizeof($duomenys);
for ($x=0;$x < $d_size; $x++)
{
// kiekviena gabaliuka skaidom pagal ','
$vars = explode(",",$duomenys[$x]);
if (isset($vars)&&sizeof($vars)==2)
{
// jei tokio variablo skripte dar nera, mes ji sukuriame
isset($GLOBALS[$vars[0]])?true:$GLOBALS[$vars[0]] = $vars[1];
}
}
}
}
decode_url();
// Shtai ir viskas :)
?>
```

Manau dabar supratote kaip veikia štai šis url rewraitingo būdas.

Apache alternatyva

Apache turi taip pat gerą modulį: mod_rewrite. Panšų url rezultatą galime gauti ir pasinaudojus apach'o mod_rewrite moduliu. Štai jums mažas pavizdys:

.htaccess'e

```
RewriteEngine On
```

```
RewriteRule ^archyvas/([0-9]+)/([0-9]+)/([0-9])+ archyvas.php?data=$1-$2-$3
RewriteRule ^archivas$ index.php
```

Dabar jei kreipsimes į serverį štai tokiu url': http://beta.php.lt/archyvas/2002/02/15 Servas perrašo url'ą ir kreipiasi štai taip: http://beta.php.lt/archyvas.php?data=2002-02-15

Įdomus variantas? Tikriausiai pamatę tokį adresą patikėtumėte kad serveryje yra kiekvienos dienos atskira direktorija :)

```
RewriteRule ^archivas$ index.php
```
Ši eilute redirektina vartotoją į index.php jei url'e nenurodyta pilna data.

Jei norite naudotis rewrite engin'u jums httpd.conf'e atkomentuoti štai šią eilutę:

```
LoadModule rewrite_module libexec/mod_rewrite.so
```
// (win konfiguracijoje ji atrodo šiek tiek kitaip - skiriasi keliai)

O ar IIS varototojai taip gali? :) Deja ne.

p.s. pamąstymui:
http://www.delfi.lt/news/daily/lithuania/article.php?id=716982
Kaip manote delfi savo serveryje turi daug daug daug direktorijų kiekvienai naujienų kategorijai ir subkategorijai?

Informacijos šaltiniai ir copyraitai

Kaip jau minėjau, dauguma informacijos paimta iš mistinio php_lt.zip, kurio net autoriaus nežinau. Jei jis atsirastų, tepraneša apie tai man.

1 skyriaus informacija imta iš php_lt.zip bei iš Nikolajaus Krauklio straipsnio betoje.
2,3,4 skyrių informacija imta iš php_lt.zip
5 skyriaus informacija imta iš php_lt.zip bei iš Nikolajaus Krauklio straipsnio betoje.
6,7 skyrių informacija imta iš php_lt.zip bei šiek tiek nagų prikišta ir iš mano pusės.
8 skyriaus informacija imta iš php_lt.zip bei iš Žygimanto Beržiūno straipsnio betoje.
9,10 skyrių informacija imta iš php_lt.zip
11,12,13,14 skyrių informacija imta iš Juliaus Pranevičiaus straipsnių betoje.
15 skyriaus informacija imta iš betos. Autoriai Juozas Šalna bei Nikolajus Krauklys.

Visi emailai:

Nikolajus Krauklys - webmaster@php.lt
Juozas Šalna - salna@ktl.mii.lt
Julius Pranevičius - slamstas@email.lt
Žygimantas Beržiūnas - webmaster@namukas.lt
Aš - scooox@delfi.lt

Geriausi PHP resursai:

http://beta.php.lt
http://php.tinklapis.lt
http://php.dar.lt

Skriptai

http://www.hotscripts.com

Ta proga norėčiau pareklamuoti ir www.fenix.mes.lt.

Apie ten visokį gramatinį šlamštą praneškite man.