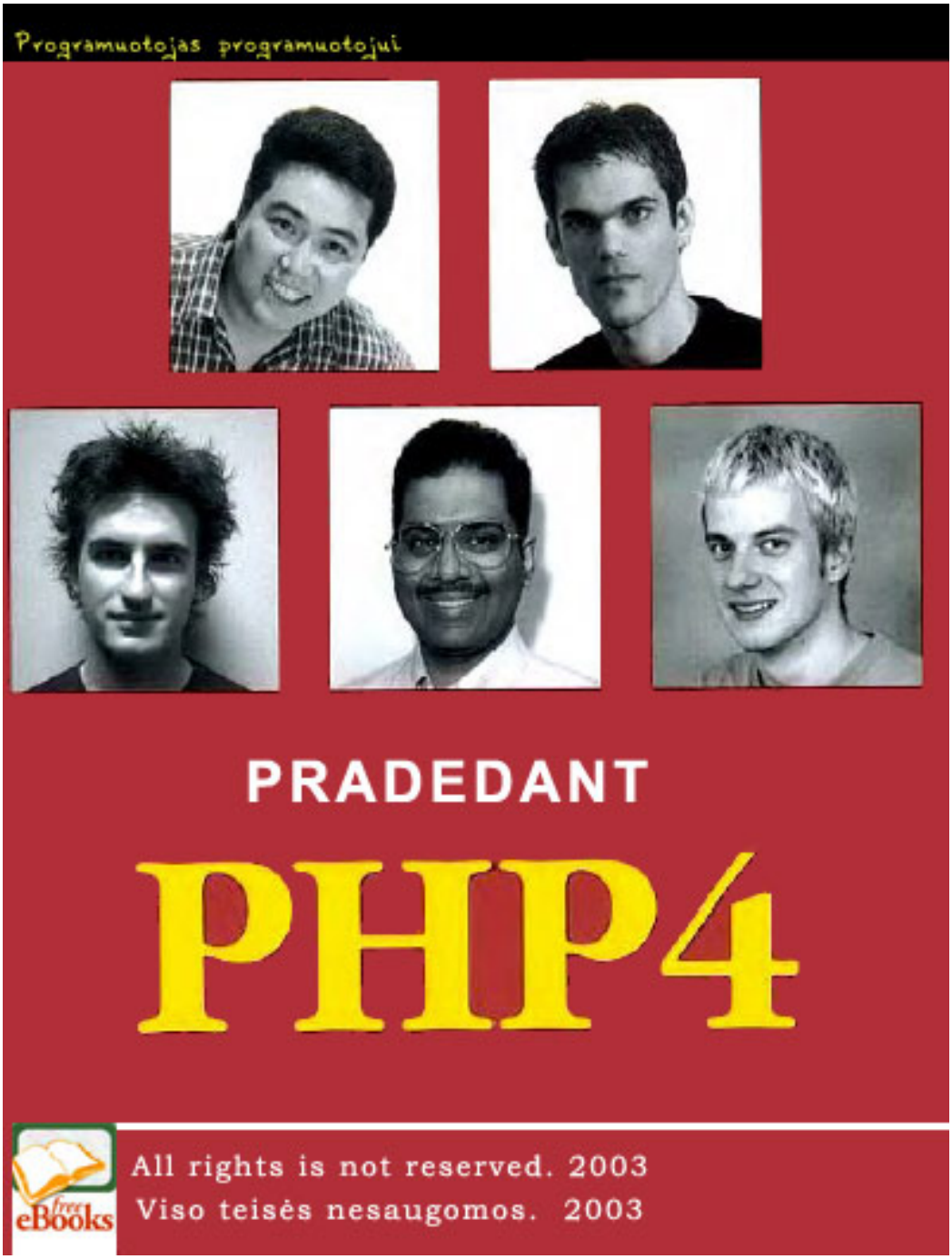


| Versta iš "Beginning PHP4" |
Kai kuri informacija yra pasenusi, nes naujausia PHP versija yra 4.3.3, tačiau pradedančiajam programuotojui jos tikrai užteks.
| Praleisti skyriai liečiantis Linux aplinką ir programavimą joje |
| Manęs tai nedomina |
| Visa kritika **NEPRIIMAMA,** jei galite padarykite geriau |

| Savo serveryje aš naudoju http://www.devside.net paketą |

Jame:

- mod_perl 1.99_09
- Apache::ASP 2.55
- mod_deflate
- PHP 4.3.3
- MySQL 4.0.15
- Perl 5.8.0
- phpMyAdmin 2.5.3
- analog 5.32
- httpd.conf, optimized and minimized, with support for php, cgi, mod_deflate, mod_perl, and Apache::ASP

Tačiau jas galite naudotis 1 skyriuje aprašytomis programomis ar atsisiųsti visus komponentus patys

Ties kiekvienu pirmą kartą verstu terminu bus rašomas jo angliškas analogas – taip stengiantis išvengti nesusipratimų dėl terminologijos

Įvadas

PHP4 yra paskutinė PHP kalbos versija – "PHP Hypertext Preprocessor". Tai programavimo kalba sudaranti galimybes konstruoti dinaminius, interaktyvius Interneto puslapius. Ją Rasmus Lerdorf sukūrė 1994 metais. Nuo to laiko ji patyrė didelius pasikeitimus ir buvo pradėta naudoti daugelio web programuotojų visame pasaulyje. Tai kas iš tiesų PHP yra?

Kalbant technine kalba, PHP4 yra nepriklausoma nuo platformos, įsiterpianti į HTML, serverio pusės, interneto scriptų kalba. Pažiūrėkime atidžiau į šiuos terminus:

❑ **Nepriklausoma nuo platformos**

Jūs galite paleisti dauguma PHP4 kodo, be apribojimų, ant kompiuterių su skirtingomis operacinėmis sistemomis. PHP4 skriptas kuris veikia Linux aplinkoje veiks ir Windows sistemoje.

❑ **Įsiterpiantis į HTML**

PHP4 kodas rašomas į failus turinčius PHP instrukcijų ir HTML kodo mišinį.

❑ **Serverio pusės**

PHP4 programos kurias mes rašysime bus vykdomos serveryje, galutinis vartotojas mato tik jų rezultatus.

❑ **Interneto skriptų kalba**

Mes naudojamės PHP4 programomis per Interneto naršyklę. Pasiekiame serverį, kuriame randasi PHP programa ir tai sužadiną pačią programą taip, kad ji tik pateikia mums savo veiklos rezultatus.

Tai reiškia – mes rašysime programas, kuriose susipynę PHP4 kodas ir HTML. Mums reikės patalpinti šias programas į web serverį norint jas paleisti. Galiausiai mes pasieksime jas per interneto naršyklę, kuri pateiks mums rezultatus HTML formate. Jūs tikriausiai jau esate susipažinę su HTML – „HyperText Markup Language". Tai kalba naudojama konstruojant interneto puslapius, savyje sujungianti paprastą tekstą ir specialius simbolius, kurie nusako naršyklei kaip vaizduoti tą tekstą. Mes naudojame HTML nustatydami, kaip skirtingi web elementai puslapyje turi būt vaizduojami, kaip puslapiai turi būt susiejami vienas su kitu, kur rodyti paveiksliukus ir t.t.

Paprastas HTML puslapis nepaisant viso savo universalumo yra ne ką daugiau nei statinis paveiksliukų ir teksto surikiavimas į reikiamas vietas, bet dauguma puslapių kuriuos jūs aplankote internete yra ne statiniai, o **dinaminiai**, netgi interaktyvūs. Jie pateikia paieškos rezultatus, praneša paskutines naujienas, ar net sveikina jus vardu kai prisijungiate prie tokių tinklapių. Šie puslapiai leidžia jums bendrauti su jais, modeliuoti juos pagal savo poreikius.

Jūs negalite sukurti tokio puslapio naudodami vien tik tai HTML, štai kada scenoje pasirodo PHP.

Mes galime suprogramuoti puslapius kurie:

❑ Pateikia duomenis iš skirtingų šaltinių – duomenų bazių ar failų

❑ Įterpti interaktyvius elementus – paieškos galimybę, skelbimų lentas ar balsavimo programą.

❑ Leisti vartotojui atlikti veiksmus – tokius kaip laiškų siuntimas ar ko nors pirkimas.

Tariant kitais žodžiais, PHP gali būti panaudota darant tokio tipo web puslapį, kokį jūs esate įpratę matyti kas dieną.

Pradedant e-komercijos tinklapiais, paieškos ir informacijos portalais, dauguma pagrindinių Internetių web puslapių savyje turi PHP kodų ar yra pagrysti vien tik

tai šiais kodais. Šioje knygoje mes, neskaitant kitų, išmoksime daryti tokias programas:

- ❑ Internetinį teksto redaktorių, kuris leis jums kurti ir redaguoti failus esančius serveryje iš bet kurio kompiuterio
- ❑ Interneto puslapį vaizduojanti parduotuvių alėją, kuris leis jums ieškoti parduodamas prekes ir vaizduoti parduotuvių išsidėstymą žemėlapyje
- ❑ Internetinį žodžių žaidimą
- ❑ Paieškos sistemą
- ❑ Konferencijų tipo elektroninio pašto sistemą, kuri leis pasirašyti į skirtingas kategorijas, o administratoriams leis siųsti laiškus visiems užsiregistravusiems.

Taigi PHP gali būt naudojama labai plačiu spektru, pradedant teksto redagavimo programomis ir baigiant galingais portaliniais interneto tinklapiais, tokiais kaip e-komercijos ar paieškos.

Kodėl PHP?

Viena iš geriausių PHP savybių yra ta, kad ją palaiko daugelis serverių ir Internet Service Providers (ISP), kas leidžią vieną kartą parašytą programą lengvai patalpinti į serverį ir visi ją galės naudotis.

Ko man prireiks?

Mes susikoncentruosime PHP naudojime Windows platformoje. Kaip pamatysime iš pirmo skyriaus mes galime suinstaliuoti PHP Windows 95 ir 98, bei NT, 2000, XP.

Be pačios PHP funkcijų bibliotekos mums reikės teksto redaktoriaus, jų sąrašą apžvelgsime pirmame skyriuje.

Jums bus reikalingas web serveris. Windows vartotojai gali naudoti Apache, ar kaip alternatyvą Microsoft'o Personal Web Server (95 ir 98 versijos) ar Internet Information Server (NT, 2000, XP versijos). Pirmas skyrius paaiškins kaip jį suinstaliuoti ir paleisti.

Norėdami gauti pilną supratimą jums reikalingas ir Interneto ryšys. Bet jei jo neturite – nepanikuokite. Jei turite įdiegtas visas reikalingas programas, daugumą pavyzdžių paleisite ir ant savo kompiuterio kuris vienu metu bus ir *serveris ir klientas*.

PHP resursai

Norėdami gauti daugiau informacijos visų pirma apsilankykite oficialiame PHP tinklapyje kurį rasite adresu www.php.net. Jis pateikia ne tik naujienas, failus, bet ir pilną dokumentaciją.

PHP4 yra pagrįstas Zend skriptų varikliu, kuris priklauso Zend Technologies, jų puslapis randasi www.zend.com. Čia rasite informaciją apie PHP4, tai pat straipsnius, studijas ir naujienas apie PHP naudojimą.

Kitas labai vertingas šaltinis yra www.phpbuilder.com puslapis, forumas PHP programuotojams. Tai naudinga vieta kur rasite patarimus ir pamokas, bei tai kas vyksta PHP programuotojų bendruomenėje.

Lietuviškas puslapis skirtas PHP – www.php.lt

1
Įdiegimas

Norėdami paleisti šioje knygoje išdėstytus programinius kodus jums reikia bent jau šių programų:

Serverio programos

- ❑ Suderinamas su PHP web serveris
- ❑ PHP4
- ❑ Daugumai pavyzdžių iš 11 skyriaus, duomenų bazė

Kliento programos

- ❑ Interneto naršyklė
- ❑ Teksto redaktorių - Notepad, Emacs, vi, BBEdit, ir t.t.

Jūs galite paleisti visas šias programas viename kompiuteryje kol mokotės. Jei turite priėjimą prie keletos kompiuterių galite serverio pusės programas įdiegti viename ir naudokis kitu kaip klientu. Šioje knygoje mes pagal nutylėjimą turime galvoje, kad visos programos randasi viename kompiuteryje. Tai konfigūraciją kuria naudoja dauguma Interneto programuotojų.

Nepanikuokite jei neturite web serverio. Dauguma neturi. 1 Skyriuje mes ruošiamės paaiškinti kaip įdiegti web serverį paprasčiausiame personaliniame kompiuteryje.

Jei jūs nerimaujate dėl Interneto ryšio, tai kompiuteris nebūtinai turi būti prijungtas prie jo, gali nebūti netgi vietinio tinklo norint paleisti serverį. Jei jūs suinstaliavote web serverį kompiuteryje visada yra galima pasiekti jį per naršyklę esančią tame pačiame kompiuteryje, net jei jūs neturite tinklo plokštės ar modemo. Žinoma norint atsisiųsti ir suinstaliuoti jums reikiamą programinę įrangą, jums bus reikalingas Interneto ryšys. Bet jums jo nereikia vien tam, kad aktyvuoti web serverį.

Kai jau turėsite web serverį, mes suinstaliuosime PHP, reikės atlikti nedidelę konfigūraciją ir mes parodysime kaip tai daroma.

Kaip ir web serveris ar PHP mums bus reikalinga ir duomenų bazė kai kuriems mūsų pavyzdžiams. Viena iš pačių galingiausių web programavimo įrankių galimybių yra galimybė pateikti informaciją saugomą duomenų bazėje žmonės naršantiems po jūsų puslapį. Visi pavyzdžiai pateikti 11 skyriuje, kurie naudoja duombazę, yra parašyti darbui su populiaria **nemokama** buombazę MySQL. Tačiau Windows ar Linux vartotojams, MySQL įdiegimas gali būti kiek skausmingas ir sudėtingas, tad bus daug parasčiau jei jūs leisite pavyzdžius vien tik suinstaliave duomenų bazę MySQL. Pilnos instrukcijos yra pateiktos 11 skyriaus pradžioje.

Tad šiame skyriuje mes paaiškinsime kaip įdiegti pačią paprasčiausią sistemą, kuri leis jums kurti savo ir paleisti pateiktus su šia knyga pavyzdžius.

Kur man pradėti?

Yra pateikti 2 būdai kuriuos galite pasirinkti priklausomai nuo jūsų sistemos:

- ❑ PHP4 įdiegimas į Windows 95 ar 98 su Microsoft Personal Web Server
- ❑ PHP4 įdiegimas į Windows NT4 ar 2000 ar XP su Microsoft Internet Information Server

PHP gali būt suinstaliuotas į daugelį skirtingų web serverio/OS kombinacijų, tame tarpe ir naudojant Apache kartu su Windows. Bet šios dvi sistemos kurias mes pasiūlėme yra paprasčiausios darbui.

PHP4 įdiegimas į Windows 95 ir 98

Visų pirma įdiekime Microsoft Personal Web Server (PWS). Reikia pažymėti, kad PWS yra tinkamas dirbti tik jūsų programuotojo kompiuteryje – šitas serveris neskirtas darbui su tikru tinklapiu. Tačiau ši sistema pateikia pakankamai gerą aplinką kuri leis moduliuoti ir programuoti kaip tikrame serveryje. Vienas iš esminių PWS trūkumų, kad šį serverį gana sunku tinkamai suinstaliuoti ir priversti korektiškai jį dirbti. Pabadykime tai padaryti sekančiais trimis žingsniais:

PWS gavimas

PWS versija kurią jums reikia suinstaliuoti yra 4.0, ji pirmą kartą išleista kartu su NT 4 Option Pack 1997 pabaigoje kaip IIS 4.0 dalis. Serveris pasiekiamas keleta būdų.

Visual InterDev

Microsoft'o Visual InterDev 6.0 versija turi savyje PWS. Serveris gali būt įdiegtas instaliuojant patį VID ar vėliau kaip pasirinkimas ir papildomų funkcijų.

Windows 98

Windows 98 CD turi PWS instaliacinį paketą. Dauguma žmonių kurie įsidiegė PWS iš Windows 98 CD į Windows 98 sistemą turi mažiau problemų nei įsidiegę iš kitų šaltinių.

FrontPage

FrontPage, FrontPage 97, FrontPage 98, FrontPage 2000, FrontPage XP turi PWS.

FrontPage 97 turi PWS 1.0, o FrontPage 98 turi PWS 4.0, dabartinę PWS versiją.

Kurį šaltinį pasirinkti

Mažiausiai problemų turėsite įsidiegę PWS iš Windows 98 CD. Vartotojams kurie iki šiol naudoja Windows 95 labai patartina pereiti prie naujesnės ir daugiau galimybių turinčios Windows98.

Įdiegimas į Windows 98 iš Windows 98 CD

Tai saugiausias būdas įdiegti PWS, bet jis veiks tik tada jei turite Windows 98 sistemą

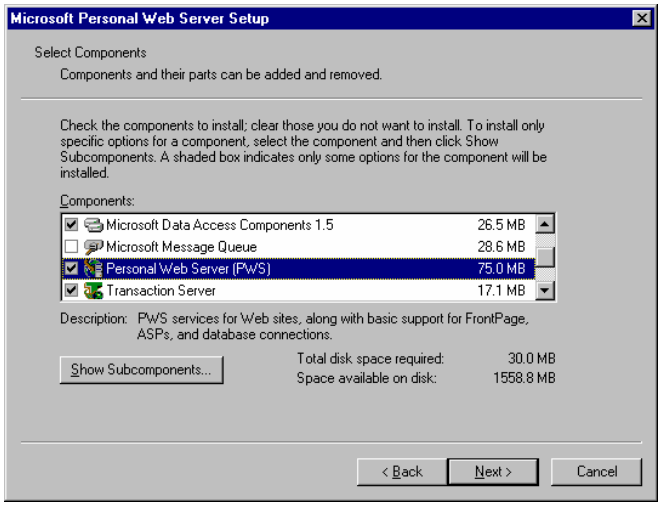
1. Įsitikinkite, kad Windows 98 CD yra diskasukyje. Kai įdėsite CD atsiras langas, pasirinkite Browse this CD.
2. Windows Explorer lange pasirinkite katalogą pavadinta `add-ons`. Jo viduje rasite katalogą `pws`. Atidarykite jį.
3. Paleiskite programą setup.exe.

PWS įdiegimas

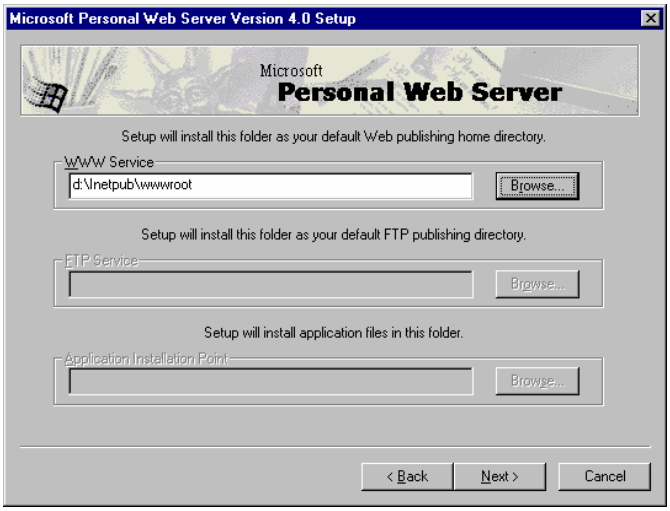
Paleidę setup programą jūs pasveikins tokio tipo langas:

Daugumai vartotojų tipinis įdiegimo būdas bus gerai. Jei pasirinksite daryti suasmenintą įdiegimą (custom install), įsitikinkite, kad bent jau pažymėjote sekančius komponentus:

- Common Program Files
- Personal Web Server



Jūsų paklaus katalogo kuriame bus saugomi web serverio failai. Normaliai pasirenkama C:\Inetpub\wwwroot, nors jūs galite pasirinkti kur tik norite. Vėliau šiame skyriuje jei męs sakysime C:\Inetpub\wwwroot, jūs turite prisiminti katalogą kurį nurodėte.



PHP įdiegimas

Dabar kai jau turite susiinstaliavę PWS, ir jūsų kompiuteris buvo paleistas iš naujo, įdiegime PHP. Visų pirma reikia jį atsisiųsti. Normaliomis sąlygomis naujausią PHP versiją galite atsisiųsti iš www.php.net, deja šiame puslapyje randasi tik pagrindinis distribucinis paketas kuris nepalaiko visų mums reikalingų funkcijų. Vietoj to geriau apsilankykite http://php4. win.de ir atsisiųskite jų vėliausią, neoficialų, stabilų paketą. Jūs atsisiųsite ZIP failą, išsaugokite jį kur nors savo kompiuteryje. Sukurkite katalogą savo PHP programoms – kažką panašaus į C:\php ir išskleiskite ZIP kataloge esančius failus.

Dabar katalogas turi keletą pakatalogių ir keletą tekstinių failų, taip pat programą `php.exe`, kuria mes tiesa pasakius tiesiogiai nesinaudosime bei, bibliotekos failą `php4ts.dll`. Jums reikia nukopijuoti šitą .dll į jūsų `\Windows\System katalogą`. Dabar atidarykite `dll` pakatalogį ir nukopijuokite ***visus*** ten esančius failus į `\Windows\System` katalogą.

Dabar grįžkite į savo PHP katalogą, ten rasite failą `php.ini`. Nukopijuokite jį į `\Windows`, ir atidarykite jį su Notepad programa. Tekste suraskite panašų tekstą:

```
  extension_dir = C:\php\extensions ; directory in which the
loadable extensions
  (modules) reside
```

Dabar, įsitikinkite, kad tai teisingas kelias iki jūsų PHP katalogo. Jei taip nėra pakeiskite į teisingą. Kitas skyrius sako PHP kokias programos dalis pakrauti. Jūs turite uždėti kabliataškius prieš visų linijų pradžią, kurios pakrauna dalis, kurių mums nereikia. Jūs galite uždėti kabliataškius prieš jas visas, išskyrus `extension=php_gd.dll`, taigi mes turėsite tokį tekstą:

```
  ;extension=php_filepro.dll
  extension=php_gd.dll
  ;extension=php_dbm.dll
  ;extension=php_mssql.dll
```

Tai reiškia, kad mes galėsime naudotis GD bibliotekos funkcijomis. Jos leis mums generuoti paveikslėlius, naudojantis PHP programomis. Kaip tai padaryti mes pamatysime 16 Skyriuje. Dabar turėtumėte išsaugoti php.ini failą.

Dabar, vėlgi su Notepad, jums reikia sukurti naują failą ir parašyti sekanti tekstą:

```
  REGEDIT4
```

```
  [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\w3svc\
parameters\Script Map]
  ".php"="C:\\php\\sapi\\php4isapi.dll"
```

Pastaba. Šitas skriptas yra įtrauktas į pavyzdžių archyvą prie 1 skyriaus kaip `PWS-php4.reg`.

Jei jūs PHP katalogo nepadarėte `C:\php`, jums reikės paredaguoti failą taip, kad Windows žinotu kur randasi jūsų `php4isapi.dll` failas. Jei jūs PHP katalogą padarėte pavyzdžiui `E:\Stuff\php4`, tada adresas bus `E:\Stuff\php4\sapi\php4isapi.dll`. Dėl to kaip šitas failas bus interpretuojamas jums reikia sudėti dvigubus įžambius brūkšnius: `C:\\php\\sapi\\php4isapi.dll`. Pavyzdžiui:

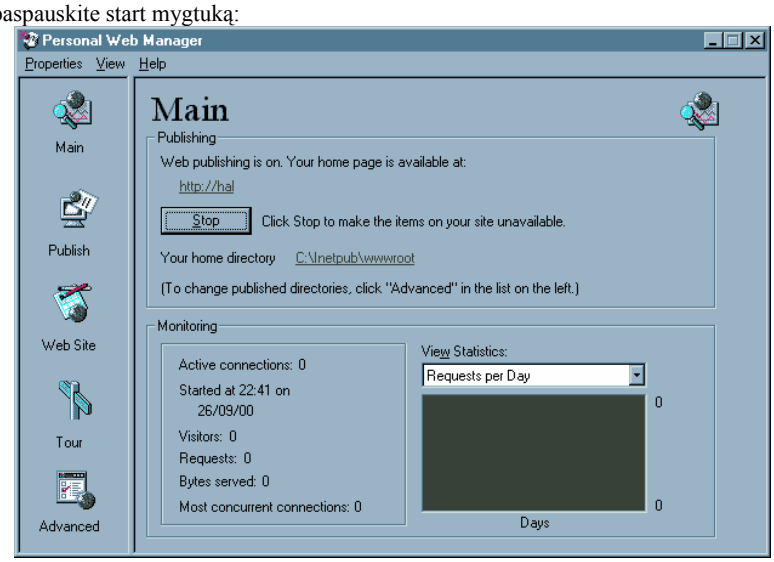
```
  REGEDIT4

  [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\w3svc\
parameters\Script Map]
  ".php"="E:\\Stuff\\php4\\sapi\\php4isapi.dll"
```

Išsaugokite šitą failą kaip, kaip `PWS-php4.reg`, uždarykite Notepad, ir du kart paspauskite ant `PWS-php4.reg`. Jūs gausite dialogo langelį klausiantį ar norite atlikti pakeitimus registre, paspauskite Yes.

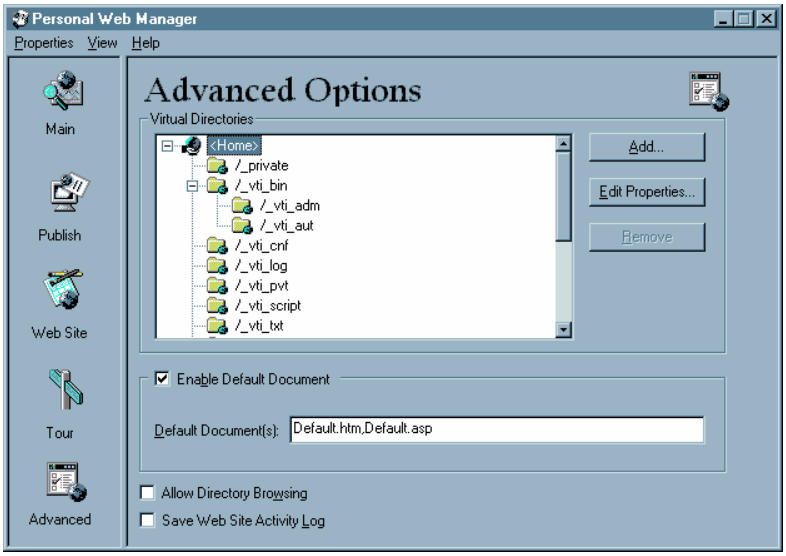
Įdiegimo užbaigimas

Dabar jums reikia paleisti Personal Web Manager, tai grafinė programa naudojama konfiguruoti PWS.

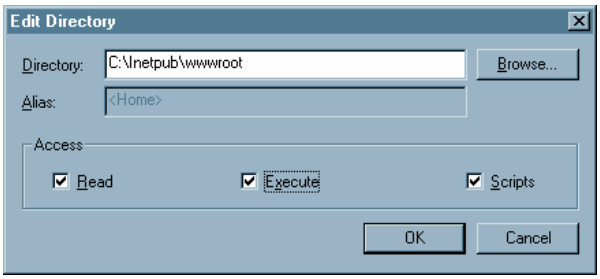
Jūs ją rasite Start/Programs/Accessories/Internet Tools/Personal Web Server/Personal Web Manager. Kai ją paleisite įsitikinkite, kad PWS paleistas. Jei taip nėra paspauskite start mygtuką:



Pasirinkite Advanced iš kairėje esančio meniu. Sekančiame lange įsitikinkite, kad viršutinis punktas, <Home>, yra pasirinktas.



Paspauskite Edit Properties, ir įsitikinkite, kad Execute yra pažymėtas.



Dabar jūs sukonfiguravote PWS, kad jis veiktu kartu su PHP4 programomis. Jums reikia perkrauti kompiuterį, kad PWS pakrautu PHP komponentus.

Šakninis jūsų serverio katalogas yra, nebent jūs pasirinkote kitaip, yra, `C:\Inetpub\wwwroot`.

Atsiminkite tai – tai svarbu. Dabar galite pereiti prie *Jūsų įdiegimo tikrinimas* dalies toliau šiame skyriuje.

PHP įdiegimas Windows NT, 2000, XP sistemose

Windows NT, 2000, XP sistemose mes naudosime Internet Information Server, kuris yra industrinis Microsofto web serveris. NT ar 2000 ar XP kompiuteris turintis IIS yra pakankama aplinka norint paleisti tikrą web serverį, nors reikia įsitikinti, kad jūs suprantate ką darote jei planuojate paleisti šį kompiuterį kaip viešai prieinamą Interneto web serverį – saugumas turėtu sukelti daugiausiai rūpesčių. IIS veikianti Windows NT ar 2000 ar XP aplinkoje yra gera programuotojo aplinka.

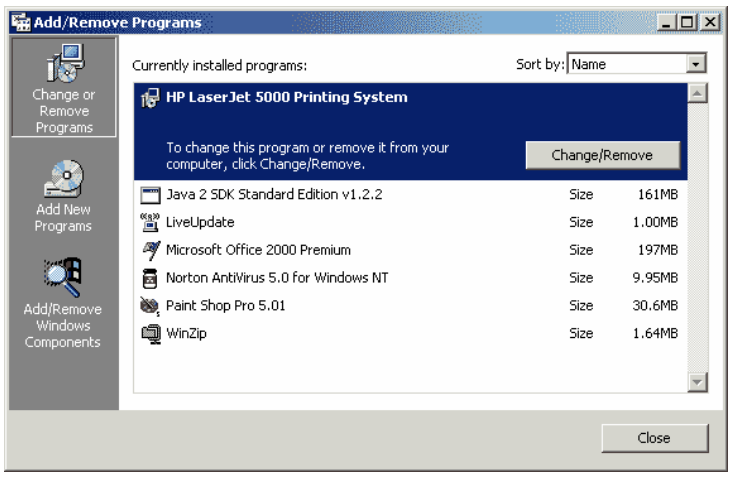
Kadangi Windows NT šiuo metu retai kur naudojama, nes dauguma kompiuterių turi Windows 2000 sistemą, čia apžvelgsime kaip tik šios sistemos įdiegimą. Windows XP sistemoje visas procesas yra iš esmės toks pat.

Internet Information Server 5.0 įdiegimas Windows 2000

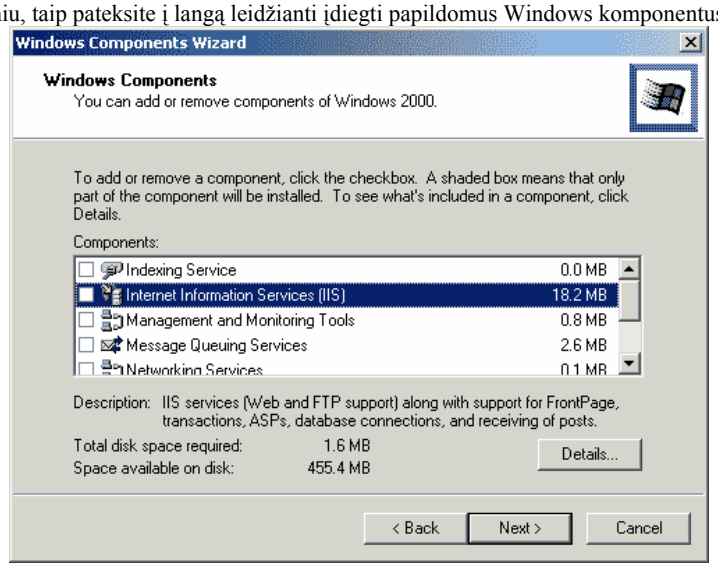
IIS 5.0 randasi Windows 2000 instaliacineme CD, taigi jis jums bus reikalingas norint įdiegti serverį.

Turėkite jį paruoštą, bet kol kas nedėkite į CD-ROM‘ą.

Pasirinkite control panel (Start | Settings | Control Panel) ir ten Add/Remove Programs ženkliuką. Atsiras sekantis dialogas rodantis įdiegtas programas:



Pasirinkite Add/Remove Windows Components punktą iš kairėje esančio meniu, taip pateksite į langą leidžianti įdiegti papildomus Windows komponentus.



Pasirinkite Internet Information Services (IIS) punktą dialogo lange, ir pažymėkite kairėje esantį langelį.

Paspauskite Next punktą, kai jūsų paprašys įdėti Windows 2000 instaliacinį CD, taip ir padarykite.

PHP įdiegimas

Dabar kai jau turite susiinstaliavę PWS, ir jūsų kompiuteris buvo paleistas iš naujo, įdiekime PHP. Visų pirma reikia jį atsisiųsti. Normaliomis sąlygomis naujausią PHP versiją galite atsisiųsti iš www.php.net, deja šiame puslapyje randasi tik pagrindinis distribucinis paketas kurie nepalaiko visų mums reikalingų funkcijų. Vietoj to geriau apsilankykite http://php4.win.de ir atsisiųskite jų vėliausią, neoficialų, stabilų paketą. Jūs atsisiųsite ZIP failą, išsaugokite jį kur nors savo kompiuteryje. Sukurkite katalogą savo PHP programoms – kažką panašaus į C:\php ir išskleiskite ZIP kataloge esančius failus.

Dabar katalogas turi keletą pakatalogių ir keletą tekstinių failų, taip pat programą `php.exe`, kuria mes tiesa pasakius tiesiogiai nesinaudosime bei, ir

bibliotekos failą `php4ts.dll`. Jums reikia nukopijuoti šitą .dll į jūsų `\Windows\System` katalogą. Dabar atidarykite `dll` pakatalogį ir nukopijuokite ***visus*** ten esančius failus į `\Windows\System` katalogą.

Dabar grįžkite į savo PHP katalogą, ten rasite failą `php.ini`. Nukopijuokite jį į `\Windows`, ir atidarykite jį su Notepad programa. Tekste suraskite panašų tekstą:

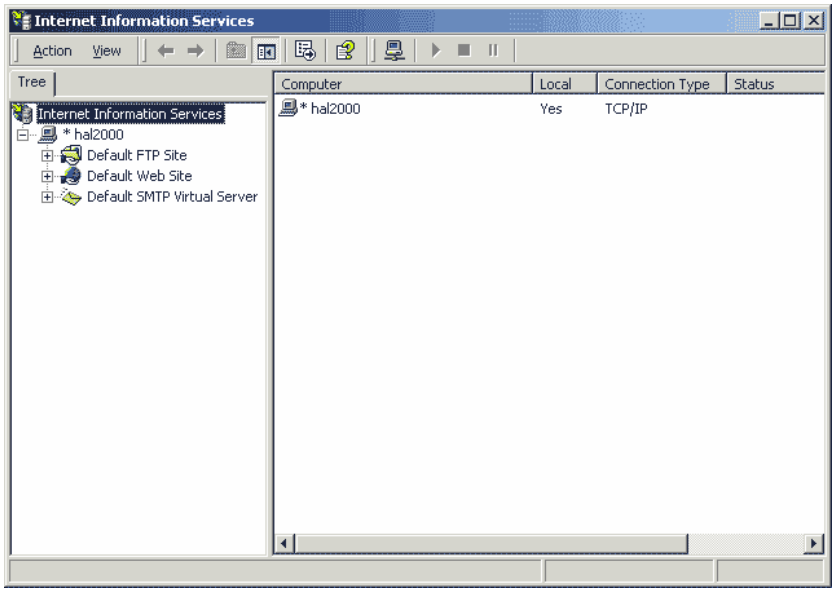
```
  extension_dir = C:\php\extensions ; directory in which the
loadable extensions
  (modules) reside
```

Dabar, įsitikinkite, kad tai teisingas kelias iki jūsų PHP katalogo. Jei taip nėra pakeiskite į teisingą. Kitas skyrius sako PHP kokias programos dalis pakrauti. Jūs turite uždėti kabliataškius prieš visų linijų pradžią, kurios pakrauna dalis, kurių mums nereikia. Jūs galite uždėti kabliataškius prieš jas visas, išskyrus `extension=php_gd.dll`, taigi mes turėsite tokį tekstą:

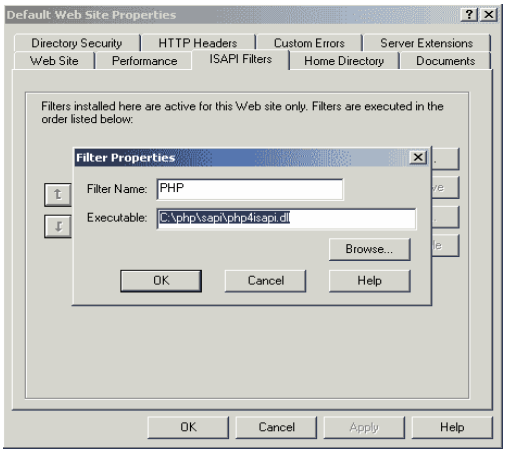
```
;extension=php_filepro.dll
extension=php_gd.dll
;extension=php_dbm.dll
;extension=php_mssql.dll
```

Tai reiškia, kad mes galėsime naudotis GD bibliotekos funkcijomis. Jos leis mums generuoti paveikslėlius, naudojantis PHP programomis. Kaip tai padaryti mes pamatysime 16 Skyriuje. Dabar turėtumėte išsaugoti php.ini failą.

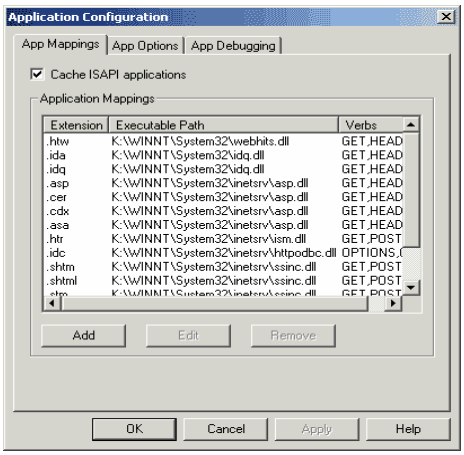
Dabar jums reikia paleisti Internet Services Manager. Windows NT, jūs rasite jį Windows NT 4.0 Option Pack jūsų Starto meniu. Windows 2000, jis bus Start | Programs | Administrative Tools. Serviso valdymas parodytas žemiau.



Dešiniu paspaudimu ant Default Web Site, jūs gausite punktą Properties. Yra du pakeitimai kuriuos mums reikia padaryti. Visų pirma mums reikia užregistruoti PHP4 ISAPI filtrą. Paspauskite ant ISAPI Filters laukelio. Paspauskite Add mygtuką, ir sukurkite naują filtrą pavadintą PHP. PHP katalogas kuriame yra mūsų atsisiųsti failai turi PHP ISAPI filtrą `sapi kataloge`, pavadinimu `php4isapi.dll`. Nurodykite teisingą kelią iki jūsų `php4isapi.dll` failo (jei jūs nuosekliai vadovavotės instrukcijomis tai turėtu būti `C:\php\sapi\php4isapi.dll`).



Sekantis dalykas kurį mums reikia atlikti tai nurodyti IIS kokius failus naudoti filtruojant PHP programas. Mes norime, kad serveris visus failus kurių galūnė yra .php skaitytu kaip PHP programas. Eikite į Home Directory laukelį, ir paspauskite Configuration.



Paspauskite Add mygtuką ir vėl nurodykite kelią iki php4isapi.dll. Nurodykite IIS pritaikyti jį .php failams, ir paspauskite OK.

Dabar mums reikia visiškai perkrauti IIS. Geriausias būdas tai atlikti yra per komandinę eilutę parašyti sekančią komandą:

> **net stop iisadmin**
The following services are dependent on the IIS Admin Service service.
Stopping the IIS Admin Service service will also stop these services.

World Wide Web Publishing Service
Simple Mail Transport Protocol (SMTP)
FTP Publishing Service

Do you want to continue this operation? (Y/N) [N]: **y**
The World Wide Web Publishing Service service is stopping.
The World Wide Web Publishing Service service was stopped successfully.

The Simple Mail Transport Protocol (SMTP) service is stopping.
The Simple Mail Transport Protocol (SMTP) service was stopped successfully.

The FTP Publishing Service service is stopping.
The FTP Publishing Service service was stopped successfully.

The IIS Admin Service service is stopping...
The IIS Admin Service service was stopped successfully.

> **net start w3svc**
The World Wide Web Publishing Service service is starting....
The World Wide Web Publishing Service service was started successfully.
>

Gale pateiktas pranešimas reiškia, kad World Wide Web Publishing servisas buvo paleistas sėkmingai ir dabar jūs turite įdiegtą serverį su PHP palaikymu. was started successfully,

Šakninis jūsų serverio katalogas yra, nebent jūs pasirinkote kitaip, yra, `C:\Inetpub\wwwroot`.

Atsiminkite tai – tai svarbu. Dabar galite pereiti prie *Jūsų įdiegimo tikrinimas* dalies toliau šiame skyriuje.

Jūsų įdiegimo tikrinimas

Gerai, mes jau turime web serverį ir PHP, reikia išbandyti kaip visa tai veikia. Atsimenate, mes prašėme prisiminti jūsų web serverio šakninį katalogą. Kaip tik dabar mums jo ir prireiks. Šakninis web serverio katalogas yra tas katalogas kur serveris ieško failų. Pavyzdžiui kai jūs surenkate URL http://www.yahoo.com/mypage.html web serveris ieškos šakniniame kataloge failo `mypage.html`. Jei toks randamas, jo turinys išsiunčiamas jums į interneto naršyklę. Panašiai, jei jūs užklausite http://www.yahoo.com/kazkas/mypage.html, web serveris savo šakniniame kataloge ieško katalogo `kazkas`, ir tada jame failo pavadinimu `mypage.html`.

Mes turime tai žinoti, nes web serveris tokias pačias taisykles taiko ir PHP programoms. Mes išsaugome savo PHP programas į failus su plėtiniu .php web serverio šakniniame kataloge ar pakatalogyje, ten pat kur serveris ieškos failų. Taigi, jei mes išsaugosime failą pavadintą `myprogram.php` www.yahoo.com web serverio šakniniame kataloge, programa bus palesta, jei į ją bus kreiptasi naršyklėje http://www.wrox.com/myprogram.php.

Bet ką mums įvesti naršyklėje, kad kreiptumėmės į savo PHP programas esančias naujai instaliuotame web serveryje. Kaip jau ankščiau buvo rašyta mes galime pasiekti web serverį per naršyklę esančią tame pačiame kompiuteryje. Kaip? Žiūrint iš kompiuterio perspektyvos vardas kurį jis naudoja kreipdamasis pats į save yra **localhost**. Taigi, jei surinksite http://localhost/ naršyklėje kompiuteris bandysi prisijungti prie web serverio kuris veikia jame pačiame. Normaliomis sąlygomis tai neveikia, nes dauguma kompiuterių neturi veikiančio web serverio. Bet kadangi mes jį jau suinstaliavome, mes galusi standartinį serverio pasisveikinimo puslapį leidžianti suprasti, kad serveris jau veikia. Jei jūsų kompiuteris prijungtas prie tinklo tai serverį galite pasiekti ir per kompiuterio vardą. Jei tiksliai nežinote savo kompiuterio vardo pabandykite pasinaudoti žemiau esančiomis komandomis:

Windows aplinkoje:

> **echo %COMPUTERNAME%**

Pavyzdžiui, jei jūsų kompiuterio vardas yra `mycomputer`, jūs galite surinkti http://mycomputer/ naršyklėje ir atsidursite tam pačiam serverio pasisveikinimo puslapyje.

Dabar mes jau pasiruošę parašyti mūsų pirmąją PHP programą ir ją išbandyti.

Bandymas – ar mūsų serveris su PHP veikia?

1. Atidarykite tekstinį redaktorių – bet kokią programą dirbančią su grynu tekstu. Notepad – idealus Windows sistemoms.

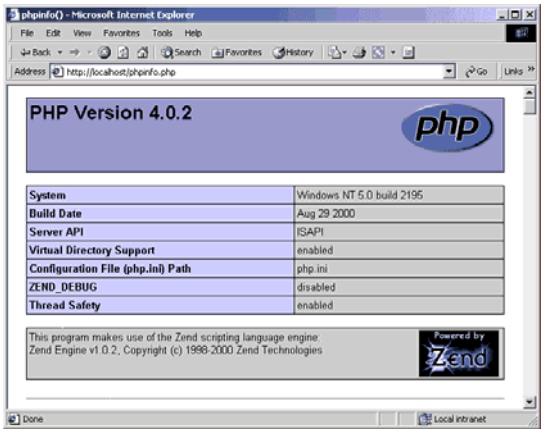
2. parašykite sekantį tekstą, *taip pat* kaip jis parašytas čia, į tuščią tekstinį failą.

```
<?php
phpinfo();
?>
```

3. Išsaugokite šitą failą savo katalogo šakniniame kataloge, kaip `phpinfo.php`. Jei jūsų tekstinis redaktorius išsaugant duodą pasirinkimo galimybę įsitikinkite, kad tekstas išsaugotas kaip plain text.

4. Dar kartą įsitikinkite, kad išsaugojot failą teisingoje vietoje su teisingu pavadinimu. Tai turėtu būti jūsų web serverio šakninis katalogas (normaliai `C:\Inetpub\wwwroot`) ir, kad jis tikrai pavadintas `phpinfo.php`. Windows aplinkoje jums reikėtu nuimti varnelę nuo punkto Hide file extensions for known file types, View tab meniu, Windows Explorer programos nuostatose Folder Options... nes kartais tai neleidžia jums įsitikinti ar tikrai failas buvo išsaugotas su teisinga galūnę.

5. Mes pasiruošę prisijungti prie mūsų serverio. Atsidarykite savo interneto naršyklę ir parašykite http://localhost/phpinfo.php. Kaip mes jau sakėme tai turėtu priversti naršyklę prisijungti prie serverio ir jame ieškoti programos pavadintos `phpinfo.php`. Jei viskas gerai, jūs pamatysite, tokio tipo vaizdą savo ekrane.



Šai taip – sveikinu jus – PHP veikia jūsų serveryje. Dabar jūs jau pasiruošę skaityti tikimesnę knygos galį. Jei jūs nematote tokio ekrano - nepanikuokite.

❑ Grįžkite atgal ir įsitikinkite, kad viską padarėte taip kaip parašyta instrukcijose. Lengva padaryti vieną lemtingą klaidą.

❑ Pakandykite išsiaiškinti kas sukėlė problemą. Ar tai web serveris? Ar PHP nustatytas tinkamai? Gal problemą sukėlė pats failas?

❑ Jei mums kyla problemos suvokiant aukščiau parašytas instrukcijas – neskaitykite knygos toliau, vis tiek nieko nesuprasite. Jei jūs nesuprantate tokių elementarių paaiškinimų, jums tikrai nesuprasti ir tolimesnių knygos skyrių kurie jau konkrečiai moko PHP programavimo kalbos.

2
Programų rašymas

Pirmame skyriuje mes pristatėme PHP ir paėjome visą reikalingų programų įdiegimo maratoną. Mes neatlikome tik vieno dalyko – nesigilinome į patį PHP kodą, paprasčiausiai patikrinome ar mūsų serveris su PHP veikia. Šiame skyriuje mes visų pirma išmoksime parašyti pati paprasčiausią PHP puslapį ir priversime jį veikti mūsų serveryje, tuo pat metu aiškindamiesi kaip jis veikia ir kokias funkcijas atlieka kodas. Vėliau mes plačiau aptarsime web serverio vaidmenį ir tiesa pasakius kaip PHP veikia jame.

Kai susipažinsime su pačiais paprasčiausiai PHP veikimo principais, mes aptarsime esminius programavimo kalbos elementus, ir kaip naudoti PHP informacijos pateikimui mūsų interneto puslapyje.

Skyriaus galas bus skirtas kintamiesiems, bei kaip jais naudotis atliekant aritmetines operacijas, ar paprasčiausią manipuliaciją su tekstu.

Šio skyriaus pagrindinės užduotys :

- ❑ Parašyti ir ištirti labai trumpą PHP programą
- ❑ Pašnekėti apie PHP variklio vietą internetiniame puslapyje
- ❑ Nustatyti ką mes turime galvoje sakydami *interpretacija* ir *vykdymas*
- ❑ Kintamieji – kas tai?
- ❑ Duomenų tipai – mes aptarsime skirtingus duomenų tipus kuriuos gali turėti kintamieji
- ❑ Kokios operacijos gali būti atliekamos su kintamaisiais
- ❑ Konstantos
- ❑ Kintamųjų vertimas iš vieno duomenų tipo į kitą
- ❑ Aplinkos kintamieji (environment variables)

PHP programos pavyzdys

Šitas skyrius prasideda pačiu paprasčiausiu vienos kodo eilutės pavyzdžiu – mes norime pademonstruoti, kad PHP puslapiai yra trijų dalykų mišinys – teksto, HTML kodo ir PHP skripto. Puslapiai turintys savyje PHP skriptus skiriasi nuo puslapių padarytu vien tik tai iš HTML, ir norint, kad juos identifikuotu PHP variklis, jie išsaugomi su `.php` (arba panašia) galūne web serveryje. Juos užklausus, puslapiai visų pirma vykdomi PHP variklyje ir rezultatai išsiunčiami vartotojui.

.php galūnė priklauso nuo jūsų esamos konfigūracijos. Jei norite PHP nustatymuose galite pakeisti labai daug dalykų, netgi PHP galūnę į pavyzdžiui .manofailas. Šioje knygoje bus naudojamos tik tai .php galūnės rašant jas mažosiomis raidėmis.

Rezultatai pateikiami naršyklėje yra HTML tipo. Mes tai greitai pamatysime, bet dabar užsiimkime pavyzdžiu.

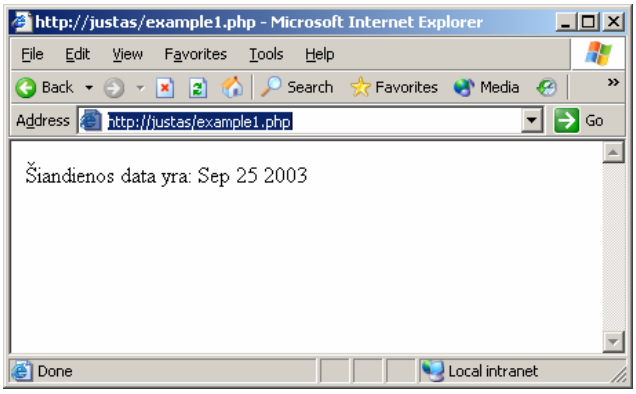
Išbandykite – Pirmoji programa

1. Atidarykite teksto redagavimo programą ir parašykit sekantį tekstą:

```
<HTML>
<BODY>
Šiandienos data yra:
<?php echo gmdate("M d Y");
?>
</BODY>
</HTML>
```

2. Išsaugokite failą kaip`example1.php` savo serverio šakniniame kataloge
.

3. Tada atsidarykite savo Interneto naršyklę ir parašykite pilną savo serverio adresą (URL) ir puslapio pavadinimą, mano būtu `http://justas/example1.php`. Jūs turite pamatyti panašų vaizdą, kaip paveiksliuke žemiau:



Skirtingi kodo tipai

Kaip mes jau sakėme šiame pavyzdyje, mes visų pirma norime pademonstruoti tris skirtingus kodo tipus naudotu šiame puslapyje. Tad pažiūrėkime į jūsų parašytą kodą ir priskirkime jį trims skirtingoms kategorijoms.

```
<HTML>
<BODY>
Šiandienos data yra:
<?php echo gmdate("M d Y");
?>
</BODY>
</HTML>
```

Kodas su baltu fonu iš tikro net nėra kodu, tai paprasčiausias **tekstas**. Ir nėra ką daugiau apie ji pasakyti.

Kodo dalis pažymėta šviesiai pilka spalva yra **HTML** tipo. Tai irgi nėra kodas tikrąją to žodžio prasme, ir jūs turėtumėte būti susipažinę su pagrindiniais HTML sintaksės elementais, tad čia mes irgi nesustosime.

Reikia pažymėti, kad šioje knygoje visas HTML kodo dalis mes rašysime didžiosiomis, nors tiesa pasakius tai prieštarauja standartams (HTML 4.0 standartai pataria visus kodus rašyti mažosiomis). Tai mes dalysime, kad išvengtume galimų nesusipratimų ir nereikėtu kiekvieną kartą naudoti foninės spalvos.

Kodas su tamsiai pilka spalva yra **PHP skriptas**. PHP skriptas atskirtas laužtiniais skliaustais ir klaustuku. Taigi, bet kada kai pamatysite kodo dalį prasidedančia `<?php ir ?>` jūs žinosite, kad viskas viduje turi būti PHP kodu.

Kaip matote iš naršyklėje pateiktų rezultatų šie tris tipai visai gerai sugyvena web puslapyje, nepaisant to fakto, kad PHP skriptas pradžioje turi būti įvykdytas serveryje, kuris gali būti ir kitas kompiuteris, o ne tik jūsų mašima.

Kaip dirba kodas

Mes jau išsiaiškinome skirtingų tipo kodus puslapyje, bet kol kas nesiteikėme paaiškinti ką daro vienos eilutės PHP kodas. Atitaisykime tai dabar – vienintelis tikras PHP skriptas puslapyje buvo šitas:

```
echo gmdate("M d Y");
```

Čia veikia trys sudedamosios dalys. Visų pirma, žodis `echo()` yra PHP komanda, kuri atvaizduoja viską kas telpą į ją internetiniame puslapyje. Tad jei mes turėtumėme komandą:

```
echo "Hello world";
```

jūs gautumėte žodžius Hello world ekrane kaip rezultatą. Tačiau kaip jūs jau pastebėjote mūsų pavyzdyje ekrane atsirado data, kadangi komandoje `echo()` mes patalpinome `gmdate("M d Y")` **funkciją** – antrąją iš sudedamųjų dalių, tuoj mes ją ir apžvelgsime.

PHP turi didžiulę biblioteką tokių rezervuotų žodžių – funkcijų – jos atlieką populiarias užduotis, tokias kaip datos gavimas, elektroninio laiško siuntimas, sudėtingų matematinių operacijų atlikimas ar skripto sustabdymas tam tikram laikui. Visų funkcijų sąrašas pateiktas Priede D. Jums nereikia jų visų prisiminti atmintinai, bet mokydamiesi mes jas vis naudosime ir naudosime.

Paskutinė dalis yra kabliataškis; PHP reikalauja, kad kiekviena kodo eilutė, (su keletu išimčių, kurias aptarsime vėliau), baigtųsi kabliataškiu.

Jūs gal būt pastebėjote, kad žodis `gmdate()` nebuvo atskirtas kabutėmis. Mūsų "Hello world" pavyzdyje, uždėdami kabutes mes liepėme PHP varikliui vaizduoti tekstą esanti viduje tiksliai tokį, koks jis yra. Mūsų atveju nedėdami kabučiu mes nurodome PHP naudoti specialią funkciją `gmdate()`, kad ši

atliktu užduotį ir gaudu datą. Ši funkcija gauną šiandienos datą ir laiką pagal Grinvičo laiko juostą (Greenwich Mean Time). Aptarkite jos veikimą plačiau.

Mūsų pavyzdyje jūs pastebėjote, kad data vaizduojama tokia:

Sep 25 2003

Tačiau yra daugybė formatų kaip ši data ir laikas gali būti pateikti. Pavyzdžiui mes galime gauti:

25/9/03 09-30AM

ar

Tuesday 25th September

Visi šie formatai teisingi ir rodo šiandienos datą, nors PHP nežino kurią versiją mes norime gauti ar gal visai kitokią datos pateikimo formą.. Taigi jums reikia parašyti, ko mes norime ir kaip tik tai daro raidės esančios tarp `gmdate()` skliaustelių. Mes nurodėme, kad mums reikia pirma mėnesio, po to dienos ir tada metų. Faktas, kad rašant mes naudojome didžiąsias raides D, M, ir Y – labai svarbus. PHP suteikia reikšmę kiekvienai raidei kaip parodyta lentelėje:

Opcija	**Rezultatas**
a	Displays "am" or "pm".
A	Displays "AM" or "PM".
d	Gives the day of the month, 2 digits with leading zeros; that is, "01" to "31".
D	Shows day of the week, textual, 3 letters; for example, "Fri".
F	Displays month, textual, long; for example, "January".
h	Shows the hour, 12-hour format; that is, "01" to "12".
H	Shows the hour, 24-hour format; that is,"00" to "23".
g	Shows the hour, 12-hour format without leading zeros; that is,"1" to "12"
G	Shows the hour, 24-hour format without leading zeros; that is, "0" to "23".
i	Displays the minutes; that is,"00" to "59".
j	Gives the day of the month without leading zeros; that is,"1" to "31".
l	Gives the day of the week, textual, long; for example, "Friday".
L	Boolean for whether it is a leap year; that is,"0" or "1".
m	Shows the month; that is,"01" to "12".
M	Gives the month, textual, 3 letters; that for example, "Jan".
n	Shows the month without leading zeros; that is,"1" to "12".
s	Displays the seconds; that is,"00" to "59".
S	English ordinal suffix, textual, 2 characters; for example, "th", "nd".
t	The number of days in the given month; that is,"28" to "31".
T	Timezone setting of this machine; for example, "MDT".
U	Displays the seconds since the epoch
w	Shows the day of the week, numeric, that is,"0" (Sunday) to "6" (Saturday).
Y	Displays the year, 4 digits; for example, "1999".
Y	Displays the year, 2 digits; for example, "99".
z	Shows the day of the year; that is,"0" to "365".
Z	Gives the timezone offset in seconds (that is,"-43200" to "43200").

Kaip matote tai gana didelis sąrašas, Vėlgi mes nesiruošiame kiekvieną kartą supažindindami su funkciją pateikti visų jos galimybių sąrašo, jums reikės žiūrėti prieduose.

Nagrinėjimo (parsing) ir vykdymo (execution) koncepcija

Jūsų PHP skripto interpretacija gali būti padalinta į du procesus. Kai puslapis patenka į serverį atsitinka du dalykai. Visų pirma puslapis yra patikrinamas ar jis teisingas programavimo kalbos atžvilgiu – šis procesas vadinamas **nagrinėjimu**. Tai analogija, kai jūsų sakinys yra tikrinamas su gramatinio tikrinimo programa

tai neužtikrina to, kad skriptas yra teisingas, patikrinama ar jis atitinka keletą iš anksto nustatytų taisyklių. Sekantis procesas vadinamas **vykdymu** atliekamas tik tada kai puslapis praeina nagrinėjimą. Štai čia pamatysite ar jūsų skriptas atlieka teisingus veiksmus. Vykdymas atliekamas imant paeiliui kiekvieną PHP eilutę ir atliekant joje nurodytus veiksmus. Visa tai daroma griežtai paeiliui, nebent PHP skripte nurodyta kitaip.

Yra dvi vietos kuriose PHP programa gali pateikti pranešimą apie klaidas – atliekant nagrinėjimą ir atliekant vykdymą. Jei taip atsitinka pranešimas apie klaidą pateikiamas naršyklei. Jei klaidų nėra dinamiškai sukurtas puslapis pateikiamas vartotojui.

Kešavimas (Cache)

Prieš pradedant gilintis į tikrą PHP programavimą, pats laikas paminėti tokios naršyklėse egzistuojančios funkcijos, kaip - **kešavimas**. Jūs gal būt jau susidūrėte su šia funkcija; ji saugo aplankytus web puslapius. Norint pagreitinti puslapių atidarymo laiką, naršyklė pateikia vartotojui seną puslapio versiją. Kešo buvimo vieta priklauso nuo naršyklės tipo ir operacinės sistemos.

Windows (98/NT ir 2000), Netscape Navigator kašas paprastai yra saugomas šioje vietoje `C:\Program Files\Netscape\Users\Cache`. Naudojant Internet Explorer jis saugomas `Temporary Internet Files` kataloge kuris randasi Windows ar WinNT katalogte Windows 98 ir NT 4.0 sistemose, bei Windows 2000 ir XP `Documents and Settings` kataloge.

Kai jūs pasitikite serveryje dinamiškai sumodeliuotais PHP puslapiais kešo egzistavimas tampa labai svarbiu dalyku. Jei jūs naudojate kešo duomenis su senesniu puslapio kodu, informacija nebus kuriama dinamiškai, o pateikiama statinė, sena puslapio versija, tad informacija gali būti pateikta neteisingai. Jūs tai pastebėsite jei pabandysite pasinaudoti naršyklės `Back` mygtuku ir grįšite prie anksčiau matyto PHP puslapio. Kartais nepakanka netgi Refresh paspaudimo, nes kešas neatnaujinamas ir naršyklė nemano, kad puslapis yra pasenęs. Šiuo atveju norint priversti Internet Explorer atnaujinti puslapio turinį reikia *CTRL+F5* kartu, o naudojant Netscape Navigator jums reikia spausti *Shift* klavišą ir spausti ant Reload.

Tai svarbu, nes paleidus pavyzdį jūs galite gauti rezultatus iš jau buvusios programos vykdymo sesijos. Jūs turite įsitikinti, kad naršyklė kiekviena kartą gauna naujausią puslapio versiją.

*Yra HTTP kodas apsaugantis nuo kešavimo. Sekančios trys kodo eilutės (naudojamos PHP skripto **pradžioje**) apsaugo Explorer ir Navigator nuo puslapio:*

```
<?
header("Cache-Control: no-cache, must-revalidate");
header("Pragma: no-cache");
header("Expires: Mon,26 Jul 1997 05:00:00 GMT");
?>
```

Kintamieji (variables)

Mes jau išnagrinėjome kaip įterpti PHP kodą į jūsų internetinį puslapį, ir kaip serveris jį supranta. Dabar laikas apžvelgti skirtingus programavimo kalbos aspektus kuriuos mes turime žinoti norint naudotis PHP. Matyt viena svarbiausi visų programinių kalbų elementai yra kintamieji. Pats paprasčiausias apibūdinimas- kad tai atminties dalelė skirta saugoti informacijai ir ši informacija priskirta konkrečiam identifikatoriui programuotojo.

PHP kalboje visi kintamieji prasideda dolerio , $ ‘ ženklu. Norėdami priskirti reikšmę kintamajam mes naudojame lygybės ženklą , = ‘ arba priskyrimo

operaciją. Taigi norint sukurti PHP kintamąjį ir priskirti jam reikšmę, jums reikia atlikti sekančius veiksmus:

```
$author = "William Shakespeare";
```

Kintamojo identifikatorius yra `$author` ir šiam kintamajam buvo priskirta reikšmė William Shakespeare. Kintamiesiems mes taip pat galime priskirti skaičius.

```
$number_of_digits_on_one_hand = 5;
```

Čia mes turime kiek sudėtingą kintamojo pavadinimą `$number_of_digits_on_one_hand`, bet išskyrus tai jis mažai kuo skiriasi nuo priskyrimo kintamajam reikšmės kurią mes padarėme anksčiau. Skirtumas tas, kad aritmetinė reikšmė (skaičius) rašomas be kabučiu. Tai nurodo PHP, kad reikšmė turi būti suprantama kaip skaičius; kintamojo tipo nurodyti nereikia.

Mums sukūrus kintamąjį, savo programoje jūs galite jį naudoti kaip tik norite. Norėdami jį pavaizduoti ekrane naudojant mūsų seną draugą `echo()` komandą, jums reikia parašyti sekantį kodą:

```
echo $author;
```

Kintamųjų vardų apribojimai

Yra keletas limitų kurie nurodo kaip jūs galite ir kaip negalite vadinti kintamųjų. Daugelyje programavimo kalbų yra kintamojo pavadinimo ilgio limitas, dažniausias 255 ar 1000 simbolių; tačiau PHP tokio limito neturi. Tačiau jums tikriausiai niekada neteks naudoti ilgesnių nei 50 simbolių pavadinimų, netgi 20-30 yra tikrai pakankamai. Pirmasis tikras griežtas apribojimas yra tas, kad kintamojo pavadinimas turi prasidėti raide ar pabraukimo ženklu , _ ‘ (ignoruojant dolerio ženklą, kuris net nėra pavadinimo dalis). Kitas apribojimas – pavadinimas turi būti sudarytas iš raidžių, skaičių ir pabraukimo ženklo. Kiti simboliai, tokie kaip +,- , *, ir & neleistini ir juos panaudojus sukels klaidą. Be šitų apribojimų jūs esate gana laisvi naudojant kintamųjų vardus. Mes vėliau šiame skyriuje aptarsime pavadinimų naudojimą.

Skiriamos didžiosios ir mažosios raidės kintamųjų varduose

Kintamieji nėra tokie paprasti kaip atrodo iš pradžių. Viena problema su kuria susiduria daugelis pradedančiųjų PHP programuotojų yra tai, kad kintamųjų varduose skiriamos didžiosios ir mažosios raidės. Paprasčiausias būdas tai paaiškinti yra pateikti gabaliuką kodo:

```
$author = "William Shakespeare";
$Author = "James Joyce";
```

Viršutinės kodo linijos iš tikro sukuria du skirtingus kintamuosius, viena pavadintą `$author` ir kitą pavadintą `$Author`. Šie du kintamieji turi dvi skirtingas reikšmes ir yra tokie pat skirtingi jei pavyzdžiui jūs naudotumėte sekantį kodą:

```
$famous_english_author = "William Shakespeare";
$famous_irish_author = "James Joyce";
```

Gana dažnai tai daro įtaką galutiniams rezultatams jei jūs netyčia panaudojate didžiąją raidę ten kur ji nebuvo naudota anksčiau. Jei jūs numatėte naudoti tik vieną kintamąjį tai kodas turėtu būti toks:

```
$author = "William Shakespeare";
$author = "James Joyce";
```

Pirmoji linija priskiria `$author kintamajam` reikšmę William Shakespeare, o antroji visiškai pakeičią kintamoji `$author` reikšmę į James Joyce. Tik tai vienas kintamasis bus sukurtas ir naudojamas šio kodo dalies.

Dabar išnagrinėkime trumpą pavyzdį kuris sukuria kintamąjį ir pateikia jo reikšmę web puslapyje.

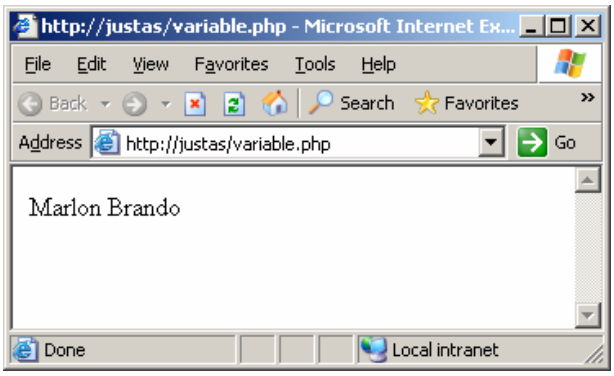
Išbandykite – Kintamojo reikšmės priskyrimas ir jo parodymas

1. Atsidarykite savo teksto redagavimo programą ir parašykit sekantį kodą:

```
<HTML>
<BODY>

<?php
$actor = "Marlon Brando";
echo $actor;
?>
</BODY>
</HTML>
```

2. Išsaugokite šitą kodą kaip `variable.php` failą.

3. Atidarykite šį puslapį savo naršyklėje.



Kaip veikia kodas

Šitas kodas susideda tik iš 2 linijų. Pirmoji:

```
$actor = "Marlon Brando";
```

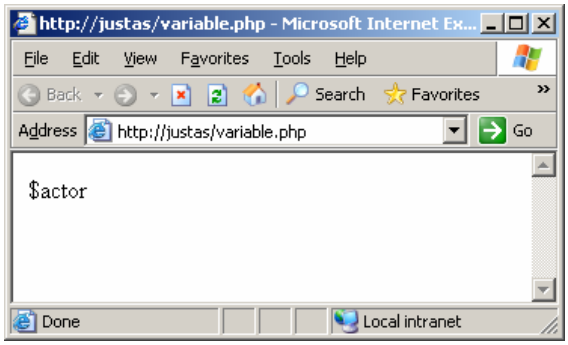
Priskiria eilutę (string) `Marlon Brando` mūsų kintamajam `$actor`. Antroji linija:

```
echo $actor;
```

pateikia kintamojo `$actor` reikšmę į `echo()` komandą. Jūs galėjote pastebėjote, kad siunčiant kintamąjį `$actor` į web puslapį mes nenaudojome kabučių. Tačiau, jei męs parašytumėme sekančia eilutę:

```
echo "$actor";
```

rezultatas *vis tiek* būtu Marlon Brando. Taip atsitinka, nes norint PHP nurodyti nenaudoti kintamojo reikšmės, o pateikti jį kaip tekstą reikia rašyti ne dvigubas, o viengubas kabutes.



Duomenų tipai

Mes jau minėjome, kad ruošiamės susikoncentruoti ant dviejų duomenų tipų, tų kurie turi skaitinę reikšmę ir tų kurie turi tekstinę reikšmę. Tačiau iš tikro yra ir daugiau kintamųjų tipų, nusakomų kaip **duomenų tipai**, kurios naudoja PHP. Pilnas sąrašas yra sekantis:

- eilutės tipas (string) //tekstas
- sveikojo skaičiaus tipas (integer) // sveikieji skaičiai
- slankaus kablelio skaičiai (double) //skaičiai su kableliu
- masyvas (array)
- objekto
- nežinomas tipas

Duomenų tipai nenurodomi programuotojo, dažniausiai pats PHP koks duomenų tipas turi būti priskirtas. Šie skirtingi tipai naudojami PHP nurodyti skirtingų tipų informaciją kuri gali būti priskirta kintamiesiems ir atlikti su jais veiksmus.

Eilutės tipas

Eilutės tipas talpina tekstinę informaciją arba žodžius ar netgi pilnus sakinius. Viskas patalpinta tarp kabučių automatiškai tampa tekstu, netgi skaičiai. Pavyzdžiui abu yra eilutės tipo kintamieji:

```
$CarType = "Cadillac";
$EngineSize = "2.6";
```

Nesvarbu, kad antroji kodo eilutė yra numeris, nes kai tik jis patenka į kabutes, jis automatiškai tampa eilutės tipo. Jei jūs priskyrėte kažkokią skaitinę reikšmę eilutės tipui, norint vėliau atlikti matematines operacijas jums reikės atlikti tam tikro tipo pavertimo operaciją.

Su eilutės tipu irgi galima atlikti tam tikrus veiksmus. Ši operacija žinoma kaip **sąryšis** (concatenation).

Eilutės sąryšis

Eilutės sąryšio procesas yra vienos eilutės pridėjimas prie kitos. Jūs galite naudoti . (tašką) kaip sąryšio operatorių, norėdami atlikti šią operaciją. Linija:

```
$Car = $CarType . $EngineSize;
```

Duos rezultatą Cadillac2.6, jei naudotume ankstesnio pavyzdžio kintamuosius ir jų reikšmes. Tiesa pasakius tarpo nebuvimas tarp dviejų eilučių gali nuvilti, tad mes turime sukurti kintamąjį kuris turės tarpo reikšmę:

```
$Space = " ";
```

Jus tikriausiai pastebėjote, kad tai skiriasi nuo tuščios eilutės turėjimo. Tuščia eilutė neturi nieko savyje, tuo tarpu eilutė su tarpu turi savyje ženklą, nors ir nematomą. Tarpas žinoma gali būtu surištas kaip ir, bet kurtis kitas tekstas:

```
$Car = $CarType . $Space . $EngineSize;
```

Tai pateiks norimą, gražų, atskirtą tekstą Cadillac 2.6.

Žinoma nėra jokios priežasties kodėl jūs negalėtumėte surišti kintamąjį su tikru tekstu:

```
$Car = "Buick" . $Space . "2.0";
```

Čia mes pridėjome kintamąjį prie teksto, kas rezultate gavosi kaip Buick 2.0. Taip pat nieko blogo neatsitiks jei mes naudosime ir sekančią eilutę:

```
$Car = "Buick" . " " . "2.0";
```

Tai duos tuos pačius rezultatus. Reikia prisiminti vieną dalyką, kad naudojant tarpus kurie vaizduojami kaip HTML jie bus pavaizduoti tik vieną kartą toj pačioj linijoj.

PHP kalboje yra dar vienas būdas surišti kintamuosius, tai mūsų seniai žinoma `echo()` komandą. Atsimenate mes anksčiau minėjome, kad jei jūs pabandysite parašyti kintamojo vardą web puslapyje jums nieko nesigaus, bus pavaizduota tik kintamojo reikšmė. Pavyzdžiui:

```
$CarType = "Cadillac";
echo "$CarType";
```

vis tiek pateiks Cadillac. Tačiau tai suteikia mums galimybę naudoti kintamąjį bei tekstą norint surišti tekstą :

```
echo "Duke's $CarType";
```

Tai pateiks Duke's Cadillac. Yra ir papildomų dalykų kurios verta prisiminti, visų pirma, kad tik naudojant dvigubas kabutes kintamojo reikšmė bus pavaizduota. Taip pat jums gali tekti susidurti su situacija kai yra du kintamieji `$Car` and `$Cars`. Kaip jūs pavaizduosite sekančia kodo eilute?

```
echo "Click here for the $Carsale";
```

Gali pasirodyti akivaizdu, nes jūs matote tai ko ir tikitės – jūs norite naudotis kintamuoju `$Car,` bet ne `$Cars`, bat kaip tai paaiškinti PHP? Tiesa pasakius PHP ieškos kintamojo pavadinto `$Carsale`. Norint išspręsti šią problemą jums reikia naudoti figūrinius skliaustelius apie kintamojo vardą:

```
echo "Click here for the ${Car}sale";
```

Jūs gal būt jau pastebėjote, kad męs visada mūsų pavyzdžiuose atitraukdavome taško simbolį nuo kintamojo. Tai ne tik gero rašymo maniera; ji lekia išvedamo teksto reikšmę. Šios dvi linijos gali atrodyti vienodai, bet jos pateikia skirtingus rezultatus

```
echo 2 . 2;
echo 2.2;
```

Atitinkamai, 22 ir 2.2! Gerai, mes truputėli apgavome jus šitoje vietoje, nes skaičiai nebuvo patalpinti į kabutes, taigi jie nebuvo eilutės tipo. Tačiau esmė vis tiek lieka teisinga, nes skaičius 2 pavyzdyje yra traktuojamas kaip eilutė sąryšio operacijoje. Taigi norint užtikrinti, kad jis nebūtu traktuojamas kaip skaičius, jums reikia pridėti tarpus tarp jo ir sąryšio operando (taško). Svarbiausia šioje pastraipoje yra prisiminti, kad tarpai gali lemti jūsų kodo veikimą ir jo pateikiamus rezultatus.

Metas pateikti pavyzdį rodantį kai kurias PHP kalbos subtilybes. Sukurkime keletą eilučių ir naudodami `echo()` komandą pateikime juos puslapyje.

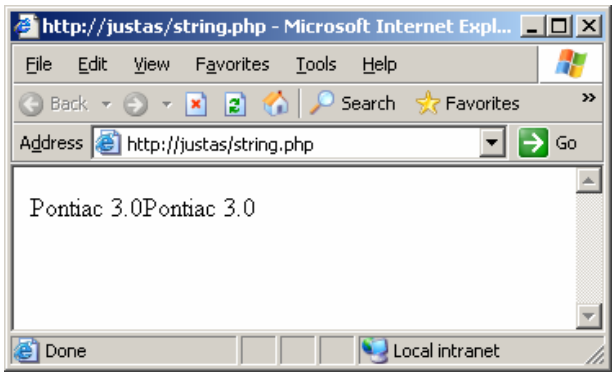
Išbandykite – Eilutės tipo naudojimas

1. Atsidarykite savo teksto redagavimo programą ir parašykit sekantį kodą:

```
<HTML>
<BODY>

<?php
$CarType = "Pontiac";
$EngineSize = "3.0";
$Space = " ";
$Car = $CarType . $Space . $EngineSize;
echo $Car . $Car;
?>

</BODY
</HTML>
```

2. Išsaugokite kaip `string.php`.

3. Atidarius jį jūsų puslapyje turėtumėte gauti kažką panašaus:



Kaip tai veikia

Mes jau prieš tai nagrinėjome labai panašų pavyzdį. Pirmosios tris kodo linijos eilutės tipo, `$CarType`, `$EngineSize`, and `$Space` ir suteikė jiems tris skirtingas reikšmes.

```
$CarType = "Pontiac";
$EngineSize = "3.0";
$Space = " ";
```

Sekančioje eilutėje mes surišome mūsų trys reikšmes, o gautą naują reikšmę priskyrėme naujai sukurtam kintamajam pavadintam `$Car`:

```
$Car = $CarType . $Space . $EngineSize;
```

Pagaliau mes atlikome sąryšį kintamojo `$Car` pačio su savimi per `echo()` komandą:

```
echo $Car . $Car;
```

Mes atsakymą pakartojome du kartus, bet be tarpo tarp dviejų kintamųjų `$Car` variables. Tai atrodo kiek negražiai, bet šitas pavyzdys buvo skirtas pademonstruoti kintamojo sąryšį pačio su savimi.

Skaitinis duomenų tipas

Yra du skaitiniai duomenų tipai, **sveikojo skaičiaus** (integers), ir **slankaus kablelio** (doubles) tipas. Štai keletas pavyzdžių:

```
$an_integer = 33;
$another_integer = -5797;

$a_double = 4.567;
$another_double = -23.2;
```

Iš pavyzdžio jūs turite suprasti, kad, bet koks sveikas skaičius automatiškai yra integer tipo, kai tuo tarpu, bet kas turintis kablelį tampa double duomenų tipo.

Šie du duomenų tipais skiriasi ir skaičių intervalo dydžiais kurios jie gali turėti. Šie dydžiai daugiausiai priklauso nuo naudojamos operacinės sistemos. Tad jei jūs turite Windows 98 PHP integer gali turėti reikšmes tarp nuo –32,768 ir iki 32,767. PHP double Windows 98 gali būti nuo –1.79769313486232E308 iki -4.94065645841247E-324 (neigiamoms reikšmėms), ir nuo 4.94065645841247E-324 iki 1.79769313486232E308 (teigiamoms reikšmėms), be to jis gali turėti ir nulinę reikšmę.

Paprastos matematinės operacijos

Yra daug matematinių operacijų kurios gali būti atliekamos PHP kalboje. Čia pateiktos pačios paprasčiausios ir dažniausiai naudojamos:

Operatorius	Operacija
+	sudėtis
*	daugyba
-	atimtis
/	dalyba
%	gaunama dalybos liekaną 8%5 =3

Jų naudojimas yra gana intuityvus. Pavyzdžiui męs galime naudoti sudėties operatorių, norėdami gauti šiandienos pirkinių kainą:

```
$Duona = 1.5;
$Pienas = 0.8;
$PirminiuSuma = $Duona + $Pienas;
```

Pirkinių sumos kintamasis yra dviejų kintamųjų – `$Duona` ir `$Pienas` suma. Jūs matote, kad `$PirminiuSuma` reikšmė bus 2.3, kas atitinka 2 litus ir 30 centų, nors programa niekur nerodo šio sąryšio su valiutą. Kitos matematinės operacijos veikia gana panašiai, tad jei jūs pirkote vien tik tai dvi duonas, programa turėtu atrodyti sekančiai:

```
$Duona = 1.5;
$PirminiuSuma = $Duona * 2;
```

Žinoma, jūs vienoje eilutėje galite rašyti ir daugiau matematinių operacijų, štai taip:

```
$Duona = 1.5;
$Pienas = 0.8;
$Nuolaida = 0.5;
$PirminiuSuma = $Duona + $Pienas - $Nuolaida;
```

Kintamojo priskyrimas prie savęs

Dar vienas dalykas kurį jūs galite daryti PHP yra sekanti kodo eilutė, kuri suglumintu matematiką:

```
$PirkiniuSuma = $PirkiniuSuma + $Duona;
```

Tiesa pasakius, tai nereiškia, kad `$PirkiniuSuma` yra lygi kintamajam `$PirkiniuSuma` plius `$Duona`, kas reiškia, kad `$Duona` lygi nuliui. Lygybės ženklas, arba priskyrimo operatorius, reiškia, kad `$PirkiniuSuma` yra priskiriama SENA `$PirkiniuSuma` reikšmė plius kintamasis `$Duona`. Tad sekanti eilutė:

```
$PirkiniuSuma = $PirkiniuSuma + 1;
```

iš tikrųjų reiškia, kad mes pridedame vienetą prie kintamojo `$PirkiniuSuma` . Yra ir sutrumpinimas kaip didinti kintamąjį vienetu ir tai atliekama štai taip:

```
$PirkiniuSuma++;
```

Kita santrupa yra:

```
$ShoppingTotal += 2;
```

ir ji atlieka tą pačią funkciją, kaip:

```
$PirkiniuSuma = $PirkiniuSuma + 2;
```

Jūs netgi galite parašyti:

```
$ShoppingTotal += $ShoppingTotal;
```

Išbandykite tai ir pažiūrėkite kas gavosi.

PHP kalboje yra ir daugiau matematinių operacijų, nei mes čia dabar aptarėme. Pradedantiesiems programuotojams yra specialus funkcijų rinkinys apskaičiuojantis trigonometrines funkcijas, logaritmus ir t.t. Šioje knygoje mes jų plačiau neaptarinėsime, visą informaciją galite rasti Priede C. Be to yra ir kitų matematinių operatorių, kuriuos mes nagrinėsime tolimesniuose skyriuose, kai jūs jau turėsite pakankamai žinių jais naudotis.

Veiksmų atlikimo seka

Šios paprastos matematinės operacijos vis tiek turi būti atliekamos pagal veiksmų atlikimo seką, kaip, kad mes naudojame matematikoje. Pavyzdžiui žemiau esanti suma gali būti skirtinga priklausomai nuo to kokia tvarka mes skaičiuosime reikšmes.

```
$Suma = 5+3*6;
```

Jei skaičiuosite šią eilute kaip ji parašyta gausite 48. Tačiau, jei naudosite matematinę veiksmų atlikimo tvarką, kur daugybos veiksmas atliekamas pirmas, o sudėtis antra, tada gausite 23. Visiškai aišku, kad reikia kažkokios taisyklės, nurodant kurie veiksmai turi būti atliekami vienas po kito.

Kaip ir matematikoje, PHP naudoja skliaustelius priverčiant atlikti matematinius veiksmus norima seka. Tad jei jūs norite, kad sudėtis būtu atlikta prieš daugybą, jums reikia parašyti skliaustelius:

```
$Suma = (5+3)*6;
```

Dabar apžvelkime nedidelį pavyzdį kuris naudoja kai kuriuos iš šių operatorių ir po nedidelių skaičiavimų pateikia rezultatus web puslapyje. Šiame pavyzdyje mes paimsime vidutine algą ir apskaičiuosime algą kai iš jos atskaičiuojama 31 % mokesčių. Po šio mokesčių atskaitymo mes dar atimsime 3% atskaitomus į pensijų fondą ir pateiksime galutinę algą, bei algą neatskaičiavus pensijai skirtų 3%.

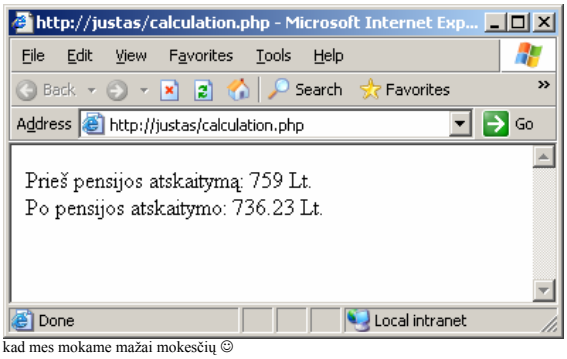
Išbandykite – Skaitiniai tipai

1. Atsidarykite savo teksto redagavimo programą ir parašykit sekantį kodą:

```
<HTML>
<BODY>

<?php
$Alga = 1100;
$Mokesciai = 31;
$Pensija = 3;
$PriesPensijosAtskaityma = $Alga - (($Alga / 100) *
$Mokesciai);
$PoPensijosAtskaitymo = $PriesPensijosAtskaityma -
(($PriesPensijosAtskaityma/100) * $Pensija);
echo "Prieš pensijos atskaitymą: $PriesPensijosAtskaityma
Lt.<BR>";
echo "Po pensijos atskaitymo: $PoPensijosAtskaitymo Lt. ";
?>

</BODY
</HTML>
```

2. Išsaugokite kaip `calculation.php`.

3 Atidarius jį jūsų puslapyje turėtumėte gauti kažką panašaus:



Ir nesakykite, kad mes mokame mažai mokesčių ☺

Kaip tai veikia

Kodas yra gana tiesmukiškas. Pirmoji linija sukuria algos kintamąjį ir suteikia jam reikšmę 1100:

```
$Alga = 1100;
```

Antroji sukuria `$Mokesciai` kurio reikšmė yra 31:

```
$Mokesciai = 31;
```

Trečioji linija sukuria kintamąjį `$Pensija` ir jo reikšmę 3:

```
$Pensija = 3;
```

Dabar męs pasiruošę atlikti kai kuriuos skaičiavimus. Visų pirma mums reikia suskaičiuoti kiek sudaro 31% nuo algos. Norint tai atlikti imame algą, ją daliname iš šimto ir dauginame iš 31. Tai duoda mums mūsų 31% reikšmę, bet dar reikia gautą reikšmę atimti iš visos algos. Tam, kad procentinė dalis būtų apskaičiuota pirma, mes įvedėme skliaustelius. Tada jau galime atimti iš visos algos ir suteikti reikšmę `$PriesPensijosAtskaityma` kintamajam.

```
$PriesPensijosAtskaityma = $Alga - (($Alga / 100) *
$Mokesciai);
```

Dabar mums reikia sužinoti kiek yra 3% nuo po-mokestinės algos, kurios reikšmė saugoma kintamajame `$PriesPensijosAtskaityma`. Norint tai padaryti mes turime atlikti tokias pačias operacijas kaip ir prieš tai, tik su

skirtingais kintamaisiais. Dabar mums reikia paimti po-mokestinę algą, padalinti ją iš šimto ir padauginti iš 3, norint gauti 3% skaičių – mes naudojome kintamąjį `$Pensija` kuriam suteikėm reikšmę 3:

```
  $PoPensijosAtskaitymo = $PriesPensijosAtskaityma -
(($PriesPensijosAtskaityma/100) * $Pensija);
```

Dabar mes turime jau abidvi reikšmes ir reikia jas išvesti į ekraną. Pastebėkite, kad mes į `echo()` komandą įterpėme HTML kodą. Lengva pamiršti, kad PHP generuoja HTML ir tai ką mes pateikiame nešyklėje bus suprasta kaip kodas.

```
  echo "Prieš pensijos atskaitymą: $PriesPensijosAtskaityma
Lt.<BR>";
  echo "Po pensijos atskaitymo: $PoPensijosAtskaitymo Lt. ";
```

Konstantos

Iki šiol mes nagrinėjome elementus kuriuos mes galėjome keisti, po to kai priskyrėme jiems reikšmes. Pavyzdžiui kintamasis `$author` kuriam mes anksčiau priskyrėme reikšmę "William Shakespeare", gali būti paleistas į "Herman Melville". Pavyzdžiui:

```
  $author = "William Shakespeare";
  echo $author "<BR>";
  $author = "Herman Melville";
  echo $author;
```

Jei patalpinsite šį kodą į puslapį `echo()` komanda pateiks du skirtingus atsakymus, priklausomai nuo kaip kito kintamojo reikšmė:

William Shakespeare
Herman Melville

Kas jei mes nenorime, kad taip atsitiktu? Kaip kurios reikšmės niekada nesikeičia, pavyzdžiui:

```
  $FreezingPointCentigrade = 0;
  $IndependenceDay = "4th July";
  $FirstPresident = "George Washington";
```

Taip pat jūs galite norėti nustatyti savo kode tokias reikšmes kurios ne tokios sukaustytos, bet vis tiek nesikeičia:

```
  $VilniausKrepsinioKomanda = "LRytas";
```

Tokiu atveju, programos vykdymo metu, jums tikrai nereikės keisti šių reikšmių ir jūs nenorėsite, kad kas nors tai darytu. Laimei, PHP turi specialius elementus kurie leidžia jums sukurti elementus kurių reikšmės negali būti pakeistos. Šie elementai žinomi kaip **konstantos**, jos leidžia sukurti kintamąjį kurio reikšmė negali būti pakeista. Būdas kaip yra sukuriamos konstantos šiek tiek skiriasi nuo įprasto kintamųjų sukūrimo būdo.

Define raktažodis

Konstantos tiesa pasakius reikalauja specialaus raktažodžio `define` norint jas sukurti. Jos taip pat neturi būti rašomos su dolerio ženklu. Tad norint sukurti konstantą kintamajam `FreezingPointCentigrade`, mes turime naudoti `define` kaip parodyta:

```
  define("FREEZINGPOINTCENTIGRADE", 0);
```

Konstantų vardai pagal susitarimą rašomi didžiosiomis raidėmis. Pirmoji reikšmė yra konstantos vardas; antroji konstantos reikšmė. Norint sukurti

konstantą kuri turi tekstą, jums tereikia apgaubti konstantos reikšmę dvigubomis kabutėmis. Taip kaip pavaizduota:

```
define("INDEPENDENCEDAY","4th July");
define("FIRSTPRESIDENT,"George Washington");
```

Tada konstantos gali būti naudojamos tokiu pačiu būdu kaip ir kintamieji. Taigi jūs galite jas pateikti puslapyje naudojant `echo()` konamdą:

```
echo "Independence Day is the " . INDEPENDENCEDAY;
```

Mes galime paversti jas į tekstą kaip viršutiniame pavyzdyje. Tačiau tas skirtumas, kad konstantos nenaudoja dolerio ženklo, tad nėra būdo kaip nurodyti PHP, kad jis atskirtu konstantą nuo paprasto teksto, tad sekanti eilutė:

```
echo "Independence Day is the INDEPENDENCEDAY";
```

parodytu tekstą Independence Day is the INDEPENDENCEDAY. Norint to išvengti jūs turite įsitikinti, kad konstantų vardai visada rašomi už kabučių. PHP taip pat turi savo konstantų kurios naudojamos norint gauti tokias reikšmes kaip operacinės sistemos pavadinimą, kurioje dirba PHP, pavyzdžiui:

```
echo PHP_OS;
```

Inicializacija

Visi turintis kiek nors programavimo patirties dirbant su kitomis kalbomis, gali būti kiek sutrikę šiame taške. Mes kalbėjome apie kintamuosius, kad daugumoje kalbų priimta pradžioje kintamuosius **paskelbti** ir **inicijuoti,** pvz. Java ar Visual Basic. Šiuose kalbose jūs turite paskelbti ir inicijuoti kintamuosius prieš pradedant jais naudotis. Paskelbimas ar iniciacija paprasčiausiai reiškia, kad prieš pradedant naudotis kintamuoju, jums reikia paskelbti, kad toks egzistuoja. Pavyzdžiui tai daroma `Dim` raktažodžiu Visual Basic kalboje:

```
Dim newVariable
newVariable = "Hello"
```

Šitas kodas nėra PHP ir veiks TIK su Visual Basic.

Tačiau PHP jums šito visko daryti nereikia. Tai nereikalaujama ir tiesa pasakius kai jūs panaudojate kintamojo vardą pirmą kartą, jis yra automatiškai sukuriamas

```
$newVariable = "Hello";
```

Tačiau PHP iniciacija turi ir vieną didelį trūkumą. Mes jau anksčiau minėjome, kad visi kintamieji turi savo duomenų tipą ir, kad PHP sukurdama kintamąjį automatiškai priskiria jam duomenų tipą. Inicijuojant kintamąjį kituose kalbose jūs dažniausiai turite nurodyti kokio tipo jis yra. Pavyzdžiui Visual Basic:

```
Dim newVariable As String
newVariable = "Hello"
```

Šitas kodas nėra PHP ir veiks TIK su Visual Basic.

PHP kalboje jūs negalite to padaryti. Tai gali privesti prie vienos problemos – kas atsitiks, jei PHP automatiškai priskirs kintamojo tipą, bet tas tipas visai ne tas kokio jūs norite? Kas atsitiks jei beskaičiuojant, pusiaukelėje, jūs užsimanysite pakeisti duomenų tipą? Pavyzdžiui, jūs galite sugalvoti leisti vartotojui įvesti informaciją, tokia kaip variklio tūris ar mašinos pavadinimas - Volkswagen Golf 2.0CL, kaip pavyzdys. Tai būtu laikoma tekstu. Tad, kas atsitiktu, jei jūs norėtumėte išgauti variklio tūri ir panaudoti skaičius atliekant matematines operacijas? Jūs turėtumėte įsivaizduoti daugelį kitų situacijų panašių į šią.

Tad, bet kokiu atveju jums reikia mokėti konvertuoti vieno tipo duomenis į kitus tipus.

Konvertavimas

PHP suteikia daug galimybių ne tik konvertuoti duomenis, bet ir nustatyti koks duomenų tipas buvo suteiktas inicijuojant kintamąjį. Tiesa pasakius daug sunkaus darbo yra panaikinama, nes PHP dažniausiai viską atlieką už tave. Dar kartą visas procesas smarkiai skiriasi nuo kitų programavimo kalbų, kurios paprastai leidžia veiksmus tik tarp keletos vienodo tipo kintamųjų. Tuo tarpu su PHP jūs galite padaryti sekančius veiksmus:

```
$EngineType = "2.0L";
$TaxRate = 3;
$TaxPaid = $EngineType * $TaxRate;
```

Tai nepakeičia nei vieno kintamojo tipo. Paleiskime šitą pavyzdį čia vietoje, kad įsitikintume kaip viskas veikia.

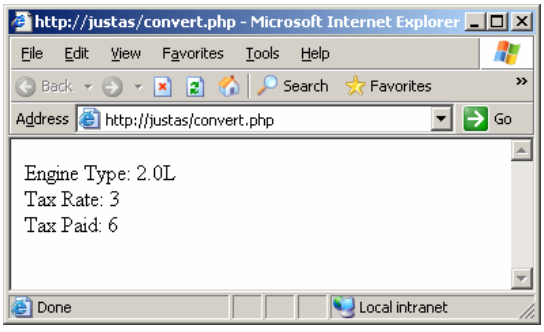
Išbandykite – Numanomas konvertavimas

1. Atsidarykite savo teksto redagavimo programą ir parašykit sekantį kodą:

```
<HTML>
<BODY>

<?php
$EngineType = "2.0L";
$TaxRate = 3;
$TaxPaid = $EngineType * $TaxRate;
echo "Engine Type: $EngineType<BR>";
echo "Tax Rate: $TaxRate<BR>";
echo "Tax Paid: $TaxPaid";
?>

</BODY>
</HTML>
```

2. Išsaugokite kaip `convert.php`

3. Atidarykite su naršykle.



Kaip tai veikia

Tai nesiskiria nuo mūsų ankstesnių pavyzdžių naudojant matematines operacijas. Pirmosios trys linijos sukuria tris kintamuosius.

Pirmasis `$EngineType` yra eilutės tipo:

```
$EngineType = "2.0L";
```

Antrasis `$TaxRate` yra skaičius:

```
$TaxRate = 3;
```

Trečioji linija atlieka kalkuliaciją, daugindama `$EngineType` ir `$TaxRate` turinį, bei patalpindama rezultatą į `$TaxPaid`:

```
$TaxPaid = $EngineType * $TaxRate;
```

Čia PHP sako, "man nerūpi, kad kintamajame `$EngineType`, yra L, aš matau skaičių ir galiu jį panaudoti daugybos veiksme“. Paskutinės trys linijos parodo, kad mūsų kintamieji gavo tokias reikšmes apie kokias mes ir mąstėme:

```
echo "Engine Type: $EngineType<BR>";
echo "Tax Rate: $TaxRate<BR>";
echo "Tax Paid: $TaxPaid";
```

Tipo keitimas

Iki šiol PHP pati keitė duomenų tipus, bet jei jūs norite naujai kuriamam kintamajam suteikti tam tikrą tipą galite tai padaryti ir patys. Viskas ką jums reikia padaryti - yra nurodyti kintamojo reikšmę ir tada nustatyti kintamojo tipą prieš jam pradedant realiai veikti. Šitas būdas vadinamas **keitimu**:

```
$NewVariable = 13;
$NewVariable = (string) $NewVariable;
```

Kodas priskyrė mūsų kintamajam skaitinę reikšmę. Tada buvo paimta skaitinė reikšmė ir paversta į tekstą antroje eilutėje. Jūs taip pat galite ir paversti duomenis atgal:

```
$NewVariable = 13;
$NewVariable = (string) $NewVariable;
$NewVariable = (integer) $NewVariable;
```

gettype ir settype

Kaip mes jau minėjome PHP turi galimybę nustatyti kintamojo duomenų tipą. Tai atliekama per `gettype()` funkciją. Kintamasis skliausteliuose pateiks savo tipą, kaip parodyta žemiau:

```
gettype($number);
```

Norint ką nors pavaizduoti ekrane, jums reikės šitą funkciją susieti su `echo()` komanda:

```
$number = 5;
echo gettype($number);
```

Tai pateiks atsakymą Integer interneto puslapyje. Yra ir panaši funkcija kuria turi PHP, vadinamą `settype()` ji panašiai kaip ir tipo keitimas, leidžia jums specialiai nurodyti duomenų tipą. Į skliaustelius įterpiamos dvi reikšmės – kintamasis kuriam norite nurodyti tipą, ir tipas kurį norite suteikti. Tai veikia sekančiai:

```
$number=10;
settype($number, "string");
```

Norint pademonstruoti, kad tai tikrai suveikė, jūs galite parodyti gautą tipą su `gettype` funkciją:

```
echo gettype($number);
```

Dabar ekrane rašoma String.

isset, unset, ir empty

Yra dar trys PHP funkcijos kurios būs naudingos dirbant su kintamaisiais. Pirmoji funkcija, `isset()`, leidžia jums nustatyti ar kažkoks kintamasis su konkrečiu vardu yra sukurtas ar ne. Jums tereikia pateikti funkcijai vieną parametrą – kintamojo vardą. Jei tai suderinti su `echo()` komandą:

```
echo isset($number);
```

Bus pateiktass numerį 1, jei kintamasi pavadintas `$number` jau egzistuoja, kitu atveju funkcija nepateiks nieko, net nulio.

Sekanti funkcija, `unset()`, skirta visiškai sunaikinti kintamąjį ir ištrinti visas su kintamuoju susijusias reikšmes. Vėlgi tereikia vieno parametro – kintamojo vardo:

```
unset($number);
```

Tačiau įsitikinkite, kad darote kaip tik tai ką norite, nes ir kintamasis ir jo reikšmė bus ištrinti.

Trečioji iš šių funkcijų, `empty()`, yra loginė priešingybė `isset()` funkcijai. Ji naudojama tokiu pačiu būdu kaip ir `isset()`, sugražindama 1, jei nėra kintamojo `$number,` arba jei `$number` lygus 0 arba "" (tuščia eilutė), ir negražinama nieko jei kintamasis egzistuoja:

```
echo empty($number);
```

Tai praktiškai užbaigia mūsų susipažinimą su kintamaisiais. Tačiau, yra dar vienas tipas kurį pateikia PHP ir apie kurį mes dar nešnekėjome.

Aplinkos kintamieji (Environment Variables)

Aplinkos kintamieji (dar žinomi kaip PHP kintamieji) yra elementai nustatyti už PHP ribų, bet pasiekiami per PHP kalbą. Šie kintamieji dažniausiai pateikia informaciją apie kliento-serverio transakcijas. Tai gali būti informacija apie HTTP užklausimus ar HTTP atsakymus ir jie turi tokį patį formatą kaip ir jūsų sukurti kintamieji. Visas skirtumas tas, kad jie buvo sukurti nepriklausomai nuo vartotojo ir nereikalauja jokio įsikišimo iš jo pusės. Jūs galite juos pamatyti naudodami `phpinfo()` funkciją kurią mes naudojomės pirmame skyriuje.

Jūs taip pat galite gauti kiekvieno jų reikšmę atskirai su `echo()` komandą:

```
echo $HTTP_COOKIE_DATA;
```

Kodas parodys, bet kokio naudojamo cookies turinį. Kitas naudingas aplinkos kintamasis yra `$HTTP_USER_AGENT` kuris nusako kliento naršyklės tipą:

```
echo $HTTP_USER_AGENT;
```

Jūs galite naudoti šia reikšmes savo programose, ar padaryti jas pritaikytas kažkokiai konkrečiai naršyklei. Kitas labai naudingas kintamasis `$HTTP_FROM` nurodantis užklausimą darančio vartotojo elektroninio pašto adresą. Jų yra ir daugiau.

Apibendrinimas

Šitas skyrius aprėpė didelį informacijos apie programavimo kalbą dalį, gana trumpoje apžvalgoje. Mes pradėjome skyrių pristatydami PHP pavyzdį, parodydami, kad PHP skriptai yra trijų dalių mišinys – teksto, HTML ir PHP kalbos. Mes pradėjome programavimo kalbos pristatymą nuo kintamųjų

koncepcijos – metodo kuriuo PHP saugo ir gauna informaciją (kaip ir dauguma programavimo kalbų). Mes išnagrinėjome skirtingus duomenų tipus ir įsigilinome į tekstinius ir skaitinius tipus. Taip pat pažiūrėjome kaip konvertuoti vieną tipą į kitą, bei pristatėme konstantas, kurios niekada nekeičia savo reikšmės. Ir galiausiai sužinojome, kad yra tokie aplinkos kintamieji, kurie tiesa pasakius net nepriklauso nuo PHP.

Verčiant šį skyrių buvo praleista dalis apie client-server ryšį, tai kaip siunčiamos užklausos į serverį ir kaip jos apdirbamos, bei HTTP pagrindai.

3

Duomenų gavimas iš kliento

Prieš tai buvusiame skyriuje mes išnagrinėjome daug praktinių pavyzdžių, priskirdami reikšmes kintamiesiems ir tada pavaizduodami kintamųjų reikšmes ekrane. Mes tiesa pasakius neprašėme vartojo įvesti jokių duomenų ir nereikalavom jokio interaktyvumo, tad iš esmės mes pateikėme jau iš anksto žinomą HTML rezultatą. Tačiau prieš tai sekęs skyrius ruošė mums dirvą, kad mes galėtumėme gauti informaciją iš vartotojo, ją išsaugoti, atlikti su ją veiksmus, bei gražinti vartotojui rezultatus kaip web puslapio dalį.

Pirmiausiai mes apžvelgsime HTML formas – tai pats populiariausias būdas kuriuo vartotojai įvedą informaciją. Tada sužinosime apie du metodus kurias HTML naudojasi perduodama informacija iš vieno puslapio į kitą. Mes sužinosime kaip PHP gali greitai perimti ir panaudoti šiuos duomenis. Yra daug metodu kaip HTML formos gali gauti informaciją, pradedant teksto langeliais, varnelėmis, slaptažodžių langeliais ir perjungimo (radio) mygtukais. Mes taip pat išnagrinėsime pavyzdžius kaip pateikti informaciją atgal šio skyriaus eigoje.

Paskutinė šio skyriaus dalis aptaria reikšmių panaudojimą, kaip atsaką į formoje įvestą informaciją, jei reikia.

Šio skyriaus pagrindinės užduotys :

- Du būdai siųsti formos informaciją – `GET ir POST`
- Teksto langeliai
- Varnelės
- Pasirinkimo mygtukai
- Sąrašo langeliai
- Paslėpti formos laukai
- Slaptažodžiai
- Submit ir Reset mygtukai
- Informacijos pagrįstos vartotojo įvestąją gražinimas

Web Formos

Tikriausiai HTML sritis sukelianti daugiausiai nesusipratimų ir klaidų yra formos. Visa tai kyla todėl, kad norint pasinaudoti duomenimis iš formos, jums reikia naudoti kokia kitą technologiją. Tai gali būti paprasta skriptavimo kalba JavaScript, ar prieštaringai vertinama Active Server Pages (ASP), CGI programa ar tikra programavimo kalba tokia Java Servlets. Norint tinkamai naudotis klientas-serveris ryšiu, negalima paprasčiausiai paimti duomenis iš vartotojo ir vėliau tokius pačius atrašyti jam atgal.

Šį skyrių mes pradėsime nagrinėdami HTML `<FORM>` reikšmę. `<FORM>` tag‘o atributai yra tiesiogiai naudojami PHP suprantant kas gi buvo pasiųsta.

FORM Tag‘as

Kas atsitinka kai jūs išsiunčiate HTML forma? Vartotojas užpildo įvairiausius laukelius ir paspaudžia siuntimo mygtuką kai yra pasiruošęs, informacija yra suformuojama vienu iš dviejų būdų ir išsiunčiama į web serverį. Tada web serveris gali ją perduoti į PHP skriptą. PHP savo ruožtu manipuliuoja šia informacija ir išsiunčia ją kaip HTTP atsakymo dalį atgal į naršyklę. Viskas ką jums reikia padaryti yra parašyti gryną HTML puslapį su atsidarančiais ir užsidarančiais `<FORM>` tag‘ais. Visi kontrolės elementai, tokie kaip teksto langeliai, varnelės, pasirinkimo mygtukai patalpinti tarp `<FORM>` tag‘ų, automatiškai tampa formos dalimi ir yra išsiunčiami į serverį.

FORM atributai

`<FORM>` tag‘as turi daugybę atributų, bet mes galime išsiversti naudodami tik du iš jų, `ACTION` ir `METHOD`.

Kiti atributai, tokie kaip `ID`, `CLASS`, `DIR`, `LANG`, `LANGUAGE`, `NAME`, `STYLE`, ir `TITLE` yra universalūs daugumai HTML tag‘ų ir nebus plačiau aiškinami. Sudėtingesnius atributus `ACCEPT-CHAR` ir `ENCTYPE`, kurie nustato kuoduotę ir MIME-TYPE formos duomenis mes specialiai nagrinėsime 10 skyriuje. Taip pat yra `TARGET` atributas, kuris panašiai kaip ir `<A HREF>` tag‘as, leidžia nustatyti kuriame langą ar frame‘ryje rodyti gautus kaip atsakymą duomenis.

ACTION

`ACTION` atributas pasako į kurį puslapį eiti kai vartotojas paspaudžia siuntimo mygtuką. Nesvarbu ar šitas puslapis bus HTML, PHP ar dar koks nors. Jis gali būti panaudotas kaip nuoroda i kitą HTML puslapį:

```
<FORM ACTION="test.html">
...
</FORM>
```

arba į PHP puslapį:

```
<FORM ACTION="test.php">
...
</FORM>
```

Tačiau kai mes nustatome, `ACTION` atribute PHP puslapį, mes iš tikro siunčiame informaciją, įvestą į formą, serveriui, kad PHP variklis galėtu su ją dirbti. `ACTION` atributas paprasčiausiai sako serveriui į kokį puslapį eiti po to – jei jūs parašysite `test.html` vietoj `test.php`, tada puslapis nebus pasiųstas į PHP ir nieko nebus parodyta nebent PHP buvo sukonfiguruota perimti .html

puslapius. Greitai mes pamatysime ką PHP daro kai gauna formą, bet prieš tai mums reikia išsiaiškinti su sekančiu `FORM` atributu `METHOD`.

METHOD

`METHOD` atributas kontroliuoja būda kuriuo informacija yra siunčiama į serverį. Kaip jau minėjome anksčiau tai gali būti atlikta dviem būdais. Tai GET ir POST metodai, juos galite naudoti sekančiai:

```
<FORM ACTION="test.php" METHOD=GET>
```

ar

```
<FORM ACTION="test.php" METHOD=POST>
```

Iš tikro yra ir daugiau reikšmių kurias galima suteikti METHOD atributui, HEAD, PUT, LINK, UNLINK, OPTIONS, DELETE, TRACE ir CONNECT. Tačiau šios opcijos naudojamos nedažnai ir mes jų plačiau neaptarinėsime.

Išnagrinėkime dvi `METHOD` reikšmes plačiau.

GET

Mes pradėsime nuo `GET` reikšmės. Tai pasako naršyklei sujungti į URL formos reikšmes kurias vartotojas įvedė. Naršyklė prideda klausimo ženklą URL gale, nurodydama kur baigiasi URL ir prasideda formos informacija. Tada formos informacija yra persiunčiama **vardas/reikšmė poromis**. Lengviau parodyti, nei plačiau paaiškinti.

Vardas/reikšmė poros veikimas labai panašus į kintamojo veikimą. Pirmoji dalis yra vardas, kuris atlieka identifikatoriaus vaidmenį. Antroji dalis yra reikšmė kuri priskirta vardui. Pavyzdžiui:

```
?animal=cat
```

Čia "animal" yra vardas, kai tuo tarpu "cat" yra reikšmė. Tai pridedama prie URL, kaip parodyta žemiau:

```
http://www.nonexistentserver.com/test.php?animal=cat
```

Naršyklė automatiškai prideda informacija prie URL kai siunčia ją į serverį. Jūs galite pridėti prie URL ir daugiau nei vieną vardo/reikšmės porą, jei atskirsite kiekvieną porą & ženklu.

Su dviem poromis URL gale tai atrodytu taip:

```
?furryanimal=cat&spikyanimal=porcupine
```

Kaip URL dalis, taip:

```
http://www.nonexistentserver.com/test.php?furryanimal=cat&s
pikyanimal=porcupine
```

Dalis prikabinta prie URL yra žinoma kaip **užklausimo eilutė** (query string). Mes jau minėjome, kad vardo/reikšmės pora yra labai panaši į kintamuosius. Tiesa pasakius kai jie patenki į PHP visada padaro juos pasiekiamus kaip kintamuosius. Tad, jei jūs pasiuntėte jūsų formą į serverį ir persikėlėte į kitą puslapį PHP padaro juos prieinamus kaip kintamuosius.

POST

Viena iš užklausimo eilutės nepatogumų yra jos viešas perdavimo būdas. Jei nenorite, kad informacija būtų rodoma URL eilutėje, jums reikės pasikliauti `POST` metodu. Jis veikia beveik identiškai kaip ir `GET` metodas; skirtumas tas, kad informacija iš formos yra siunčiama HTTP užklausimo viduje, o ne URL. Tai

reiškia, kad informacija yra nematoma visiems, nes ji neprikabinta prie URL. `POST` taip pat leidžia siųsti didesnius duomenų kiekius – yra fizinis limitas kiek informacijos gali būti pasiųsta kaip URL dalis.

HTML Formos Elementai ir PHP

Mes jau išsiaiškinome proceso kai kurias detales, tad apžvelkime dažniausiai naudojamus HTML elementus kuriuos galite naudoti rinkdami informaciją formoje, bei kaip naudoti PHP dirbant su šia informacija. Visi tolau toliau pateikiami pavyzdžiai reikalauja dviejų puslapių. Pirmojo puslapio pagalba gaunama informacija kurią pateikia vartotojas, o antrasis pasiunčia jau apdorota informacija iš web serverio ir PHP variklio atgal į vartotojo naršyklę.

Pirmasis puslapis neturi PHP kodo iš viso. Tiesa pasakius, daugelis puslapių, kurie turės formas bus sudaryti grynai iš HTML ir turės galūnes `.htm` ar `.html`. Mes stebėsime šį formatą visuose mūsų pavyzdžiuose.

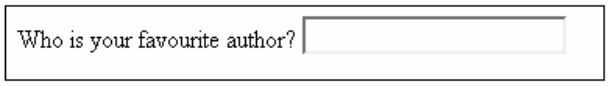
Pradėkime pažintį su populiariausiais kontrolės elementais.

Teksto langeliai (Text Boxes)

Teksto langeliai turbūt yra vieni žinomiausių su kurias jums teko susidurti klaidžiojant po Internetą. Jie sukuriami naudojant `<INPUT>` elementą ir nustatant `TYPE` atributą tekstui.

```
<INPUT TYPE="Text" NAME="TextBox1">
```

Jų privalumas tas, kad jie gali priimti visą teksto sakinį iš vartotojo. Tai daro juos idealius, kai nežinoma kiek atsakymų (simbolių ar žodžių) parašys vartotojas. Tipinis teksto laukelis atrodo taip:



Mes gana ilgai nedarėme praktinių užsiėmimų, tad pereikime prie jų. Mes paaiškinsime, kas vyksta po to kai išbandysime pavyzdį. Šiame pavyzdyje mes paprašysime parašyti vartotojo parašyti jo mėgstamiausią autorių ir parodysime jį kitame puslapyje.

Išbandykite – Teksto langelių naudojimas

1. Atidarykite teksto redagavimo programą ir parašykit sekantį HTML tekstą:

```
<HTML>
<HEAD></HEAD>
<BODY>
<FORM METHOD=GET ACTION="text.php">
Who is your favourite author?
<INPUT NAME="Author" TYPE="TEXT">
<BR>
<BR>
<INPUT TYPE=SUBMIT>
</FORM>
</BODY>
</HTML>
```

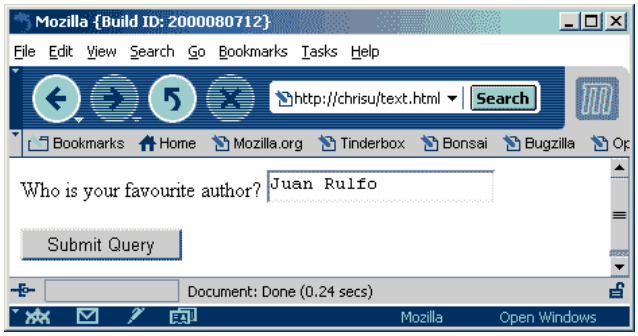
2. Išsaugokite šitą kodą kaip text.html failą.

3. Uždarykite šitą failą ir atsidarę naują, parašykite:

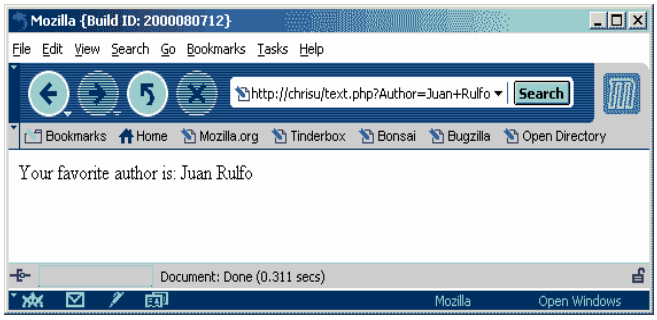
```
<HTML>
<HEAD></HEAD>
<BODY>
Your favorite author is:

<?php
echo $Author;
?>

</BODY>
</HTML>
```

4. Išsaugokite šitą kodą kaip text.php failą.

5. Atidarykite puslapį text.html savo naršyklėje, atsakykite į klausimą



6. Paspauskite ant Submit Query mygtuko ir jūs turėtumėte pamatyti jūsų parašyta vardą, taip kaip pavyzdyje.



Kaip tai veikia

Visų pirma, atkreipkite dėmesį į URL ankstesniame paveiksliuke (antras mūsų sukurtas puslapis). Prie `text.php` galo prikabinta užklausos eilutė. Tai buvo pridėta naršyklės, nes mes nurodėme tai padaryti `text.html`, faile su sekančiu kodu:

```
<HTML>
<HEAD></HEAD>
<BODY>
<FORM METHOD=GET ACTION="text.php">
Who is your favourite author?
...
```

Nustatydami atributą GET mes nurodėme mūsų formos informaciją siųsti kaip užklausimo eilutę. Išnagrinėkime pačią užklausimo eilutę. Apatiniame pavyzdyje ji rodo:

```
?Author=Juan+Rulfo
```

Mes jau sakėme, kad užklausimo eilutė susidaro iš vardas/reikšmė poros. Jums nereikia būti Sherlock Holmse, norint suprasti, kad Author yra vardas, o Juan Rulfo reikšmė šiame pavyzdyje. Užklausimo eilutė paima Author kaip vardą iš paryškinto kodo pirmame mūsų puslapyje, text.html:

```
Who is your favourite author?
<INPUT NAME="Author" TYPE="TEXT">
<BR>
```

NAME atributas <INPUT> tag'o nustato kad šio elemento vardas yra Author. Reikšmę šiam elementui mes priskyrėme parašę tekstą, šiuo atveju autoriaus pavardę tekstiniame laukelyje.

Gautoji užklausos eilutė patenka į mūsų antrąjį failą, tad išnagrinėkime ir jį. Tiesa pasakius tai tik viena teksto eilutė ir viena PHP kodo eilutė:

```
Your favorite author is:
<?php
echo $Author;
?>
```

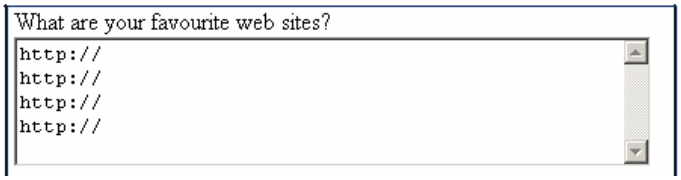
PHP komanda parodo kintamojo $Author turinį. Mūsų kode mes niekur patys nesame sukūrę kintamojo $Author. Mes tik sukūrėme teksto laukėlį ir suteikėme jam vardą Author. Kai mes pasiuntėme formą į web serverį ir PHP variklį jis pats sukūrė toki kintamąjį. Jei mes sukurtumėme laukeli pavadintą "Name", tada mūsų kintamasis būtu $Name. Tai viskas ką daro mūsų pavyzdžio programa.

Teksto laukeliai (Text Areas)

Jei norite turėti teksto lauką kuriame vartotojas galėtu įvesti keletą ir daugiau teksto linijų, jums reikės visai kitokio HTML komponento. Jūs netgi nenaudosite HTML <INPUT> tag'o; vietoj jo jūs naudosite <TEXTAREA> tag'ą. <TEXTAREA> turi kiek kitą sandarą ir jame jūs galite nurodyti kiek eilučių ir stulpelių jis turi būti. Pavyzdžiui:

```
<TEXTAREA NAME="WebSites" ROWS="30" COLS="50">
```

Kaip ir prieš tai buvęs komponentas jis skirtas priimti visą sakinį iš vartotojo. <TEXTAREA> privalimas, kad jūs galite nustatyti dydį, tad jis gali turėti keletą teksto eilučių. <TEXTAREA> reikalauja ir uždarančio tag'o, tad tarp jų jūs galite įdėti ir savo tekstą. Pavyzdžiui:



Pasižiūrėkime į kitą pavyzdį. Šiame mes darysime tą patį kaip ir ankstesniame ir pateiksime visus puslapius, kuriuos įves vartotojas. Tačiau mes padarysime vieną esminį pakeitimą perduodant informaciją, kaip tuoj patys įsitikinsite.

Išbandykite – naudojimasis teksto laukeliais

1. Dar kartą atsidarykite savo patikimą teksto redaktorių ir parašykite:

```
<HTML>
<HEAD></HEAD>
<BODY>
<FORM METHOD=POST ACTION="textarea.php">
What are your favourite web sites?
<TEXTAREA NAME="WebSites" COLS="50" ROWS="5">
http://
http://
http://
http://
</TEXTAREA>
<BR>
<BR>
<INPUT TYPE=SUBMIT>
</FORM>
</BODY>
</HTML>
```

2. Išsaugokite kaip `textarea.html`.

3. Uždarykite prieš tai redaguotą failą ir pradėkite naują:

```
<HTML>
<HEAD></HEAD>
<BODY>
Your favorite web sites are:
<?php
echo $WebSites;
?>
</BODY>
</HTML>
```

4. Išsaugokite kaip `textarea.php`.

5. Atidarykite `textarea.html` savo naršyklėje ir parašykite keletos web puslapių adresus:

6. paspauskite ant Submit Query kai jau parašėte puslapių adresus. Jūs neturite užpildyti jų visų, net gi mes palikome vieną tuščią. Panašus paveikslėlis turėtu atsirasti pas jus:

Kaip tai veikia

Ne taip gražu ir tvarkinga kaip jūsų ankstesniame pavyzdyje ar ne? Tačiau neleiskite, kad atitrauktu jus nuo svarbaus dalyko – URL kuris yra:

```
http://chrisu/textarea.php
```

Čia nėra prikabinta eilutės. Taip atsitiko todėl, kad mes pasirinkome `METHOD` būda `POST`.

```
<HTML>
<HEAD></HEAD>
<BODY>
<FORM METHOD=POST ACTION="textarea.php">
What are your favourite web sites?
...
```

Tai vienintelis dalykas kurį mums reikėjo padaryti, kad formos detalės nebūtu viešai matomos. Yra ir kitų įdomių dalykų, `<TEXTAREA>` tag'o sintaksė:

```
<TEXTAREA NAME="WebSites" COLS="50" ROWS="5">
http://
http://
http://
http://
</TEXTAREA>
```

Mes nustatėme, kad teksto laukas turi būti 5 eilių aukščio ir 50 ženklų pločio. Tekstas kitaip nei paprastame HTML nereikalauja perkėlimo į kitą eilutę (su
 ar kitaip), užtenka pradėti naują liniją ir jis vaizduojamas naujoje. Mes nustatėme, kad `TEXTAREA` vardas būtu "WebSites", ir tada sekančiame puslapyje mes pasinaudojome šiuo kintamuoju kaip `$WebSites`, kalbant dar kartą mes turime identišką situaciją, kai formos elementas tampa kintamuoju:

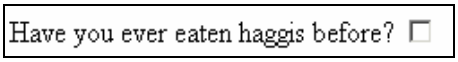
```
...
Your favorite web sites are:
<?php
echo $WebSites;
?>
...
```

Varnelės (Check Boxes)

Varnelės, kaip ir teksto laukeliai yra sukuriami HTML kalboje su `<INPUT>` tag'u. Jas sudaro vienas laukelis, kurį galima pažymėti. Tai nereikalauja jokių duomenų iš vartotojo, išskyrus paspaudimo, tad visi duomenys kuriuos turės varnelės smarkiai skirsis nuo teksto laukelių. HTML sintaksė yra panaši, skiriasi tik tipas:

```
<INPUT NAME="Choice" TYPE="Checkbox">
```

Varnelės naudojamos kai užduodamas griežtas klausimas taip/ne, nepaliekant kitų variantų:



Varnelės taip pat turi `CHECKED` atributą (kuriam nereikia suteikti reikšmės). Jei jūs varnelei parašysite šį atributą, ji bus pažymėta iš pat pradžių

```
<INPUT NAME="Choice" TYPE="Checkbox" CHECKED>
```

Tuo pačiu `VALUE` atributas yra "on" iš pat pradžių.

Visi varnelių privalumai atsiskleis tik kai jas pradėsite naudoti, tad negaišdami laiko pažiūrėkime pavyzdį.

Išbandykite – varnelių naudojimas

1. Dar kartą atsidarykite savo patikimą teksto redaktorių ir parašykite:

```
<HTML>
<HEAD></HEAD>
<BODY>
<FORM METHOD=POST ACTION="checkbox.php">
Have you ever eaten haggis before?
<INPUT NAME="Choice" TYPE="Checkbox">
<BR>
<BR>
<INPUT TYPE=SUBMIT>
</FORM>
</BODY>
</HTML>
```

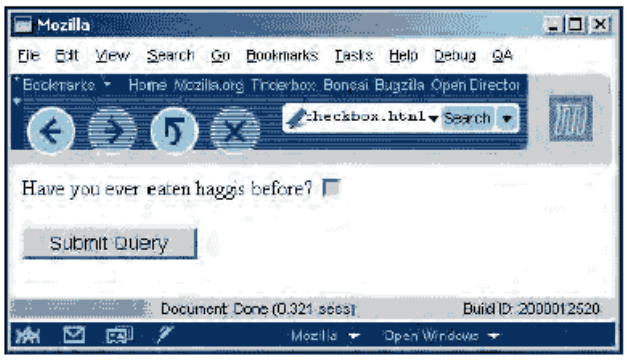
2. Išsaugokite kaip `checkbox.html`.

3. Uždarykite prieš tai redaguotą failą ir pradėkite naują:

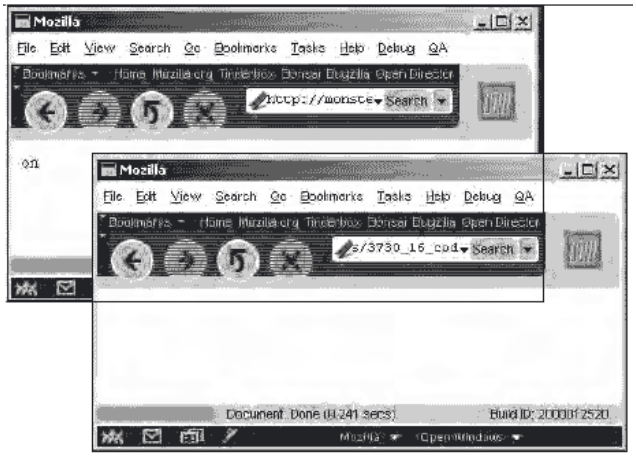
```
<HTML>
<HEAD></HEAD>
<BODY>
<?php
echo $Choice;
?>
</BODY>
</HTML>
```

4. Išsaugokite kaip `checkbox.php`.

5. Atidarykite `checkbox.html` savo pasirinktoje naršyklėje:



Priklausomai nuo to ar buvo pažymėta varnelė prieš paspaudžiant Submit Query, jūs galsite du rezultatus.



Kaip tai veikia

Jūs tikriausiai jau pripratote prie šio ritualo. Dar kartą, jei pažiūrėsite į URL – čia nėra jokių prikabintų duomenų. Taip yra todėl, kad mes `POST` metodą.

Taip nurodyta mūsų pirmame faile, `checkbox.html`:

```
<HTML>
<HEAD></HEAD>
<BODY>
<FORM METHOD=POST ACTION="checkbox.php">
```

Mūsų formos elementas yra varnelė, tad ją sukuriame su `<INPUT>` tag'u:

```
Have you ever eaten haggis before?
<INPUT NAME="Choice" TYPE="Checkbox">
```

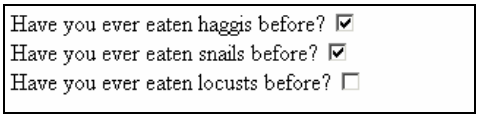
Tai viskas ką galima pasakyti apie mūsų pirmąjį failą. Antrajame, `checkbox.php`, mes naudojamės PHP kintamuoju, kuris dar kartą buvo toks pats kaip ir formos varnelės vardas:

```
<?php
echo $Choice;
?>
```

Visas skirtumas, kad dabar buvo sukurtas kintamasis su reikšme kurios mes jam nesuteikėme. Jei varnelė buvo pažymėta, jos reikšmė yra 'on'. Jei ne – tada reikšmės nėra.

Keletas varnelių

Kas atsitiks jei norėsime naudoti daugiau nei vieną varnelę toje pačioje formoje. Jei jūs susipažinę su pasirinkimo mygtukais, tai žinote, kad pasirinkdami vieną jūs tuo pačiu metu nuimate pažymėjimą nuo kitų. Varnelės dirba kitaip. Jų privalumas tas, kad kiekviena varnelė yra atskiras elementas. Tad jūs galite pažymėti keletą varnelių ra nepažymėti nei vienos. Pavyzdžiui mes galime modifikuoti mūsų prieš tai buvusį pavyzdį, kad jis atrodytu taip:



Taip ir padarykime, mes grįšime atgal ir modifikuosime prieš tai buvusį pavyzdį, taip kad jis turėtu keletą varnelių.

Išbandykite – keletos varnelių naudojimas

1. Atidarykite teksto redaktorių, pakraukite `checkbox.html`, ir pagal pavyzdį parašykite:

```
<HTML>
<HEAD></HEAD>
<BODY>
<FORM METHOD=POST ACTION="checkboxes.php">
Have you ever eaten haggis before?
<INPUT NAME="Choice1" TYPE="Checkbox" VALUE="Haggis">
<BR>
Have you ever eaten snails before?
<INPUT NAME="Choice2" TYPE="Checkbox" VALUE="Snails">
<BR>
Have you ever eaten locusts before?
<INPUT NAME="Choice3" TYPE="Checkbox" VALUE="Locusts">
<BR>
<BR>
<INPUT TYPE=SUBMIT>
</FORM>
</BODY>
</HTML>
```

2. Išsaugokite kaip `checkboxes.html`.

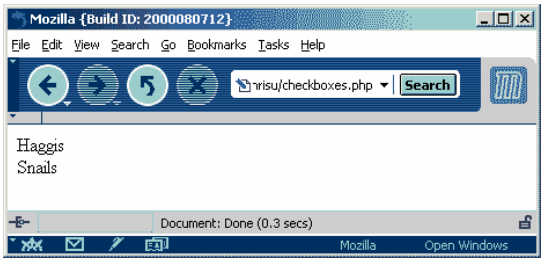
3. Uždarykite šitą failą ir atsidarykite naują:

```
<HTML>
<HEAD></HEAD>
<BODY>
<?php
echo "$Choice1<BR>";
echo "$Choice2<BR>";
echo "$Choice3<BR>";
?>
</BODY>
</HTML>
```

4. Išsaugokite kaip `checkboxes.php`.

5. Atidarykite `checkboxes.html` savo naršyklėje.

6. Paspauskite keletą pasirinkimų ir paspauskite siuntimo mygtuką, jūs turėtumėte gauti kažką panašaus:



Kaip tai veikia

Mes nustatėme `VALUE` atributą kiekvienai varnelei pirmajame faile:

```
Have you ever eaten haggis before?
<INPUT NAME="Choice1" TYPE="Checkbox" VALUE="Haggis">
<BR>
```

```
Have you ever eaten snails before?
<INPUT NAME="Choice2" TYPE="Checkbox" VALUE="Snails">
<BR>

Have you ever eaten locusts before?
<INPUT NAME="Choice3" TYPE="Checkbox" VALUE="Locusts">
```

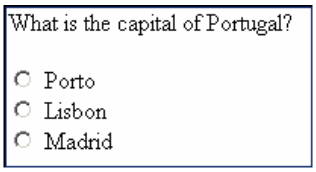
Tai duodą rezultatą kiekvienai varnelei jei ji buvo pažymėta. Tad jei `Choice1` varnelė buvo pažymėta, jai suteikiama `Haggis` reikšmė (vietoj standartinio 'on'), ir šią reikšmę mes perduodame `$Choice1` kintamajam `checkboxes.php` puslapyje. Jei varnelė nebuvo pažymėta tai nieko nėra perduodama ir kintamasis negauna reikšmės. Antrame faile, `checkboxes.php`, mes parodysite šių kintamųjų turinį, štai taip:

```
echo "$Choice1<BR>";
echo "$Choice2<BR>";
echo "$Choice3<BR>";
```

Iš ekrano nuotraukos jūs turbūt supratote, kad Haggis ir Snails varnelės buvo pažymėtos, Locusts ne.

Pasirinkimo mygtukai (Radio Buttons)

Pasirinkimo mygtukai yra savotiški varnelių pusbroliai. Jei jūms reikalinga pasirinkimo galimybė, bet reikia tik vieno galimo atsakymo, turėtumėte naudoti pasirinkimo mygtukus. Pavyzdžiui klausimas su keletu atsakymo variantų:



Vėlgi pasirinkimo mygtukai yra sukuriami su `<INPUT>` tag'u, nustatant `TYPE` atributą kaip Radio.

```
<INPUT NAME="Question1" TYPE="Radio" VALUE="Porto">
```

Pasirinkimo mygtukai kaip ir varnelės turi atributą `CHECKED`. Jei jūs suteiksite šį atributą, pasirinkimo mygtukas bus pažymėtas iš pat pradžių:

```
<INPUT NAME="Question1" TYPE="Radio" CHECKED>
```

Jei nesuteisite `VALUE` atributo, pasirinkimo mygtuko reikšmė bus "on".

Norint sujungti grupę pasirinkimo mygtukų, visiškai priešingai nei varnelėms, reiia suteikti vienodus vardus. Pavyzdžiui:

```
<INPUT NAME="Question1" TYPE="Radio" VALUE="Porto">
<INPUT NAME="Question1" TYPE="Radio" VALUE="Lisbon">
<INPUT NAME="Question1" TYPE="Radio" VALUE="Madrid">
```

Šiuo būdu jūs sakote web serveriui, kad šie pasirinkimo mygtukai yra susyją. Jei suteikti jiems skirtingus vargus jūs galėsite žymėti juos visus, kas prieštarauja pačiai jų paskirčiai.
Pabandykime atlikti pavyzdį su sekančių kodu.

Išbandykite – pasirinkimo mygtukų naudojimas

1. Atidarykite savo redaktorių ir kaip tikriausiai atspėjote – parašykite sekantį kodą:

```
<HTML>
<HEAD></HEAD>
<BODY>
<FORM METHOD=GET ACTION="radio.php">
What is the capital of Portugal?
<BR>
<BR>
<INPUT NAME="Question1" TYPE="Radio" VALUE="Porto">
Porto
<BR>
<INPUT NAME="Question1" TYPE="Radio" VALUE="Lisbon">
Lisbon
<BR>
<INPUT NAME="Question1" TYPE="Radio" VALUE="Madrid">
Madrid
<BR>
<BR>
<INPUT TYPE=SUBMIT>
</FORM>
</BODY>
</HTML>
```

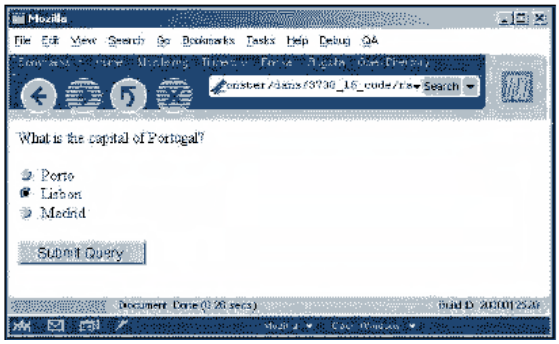
2. Išsaugokite kaip `radio.html`.

3. Uždarykite šitą failą ir sukurkite naują:

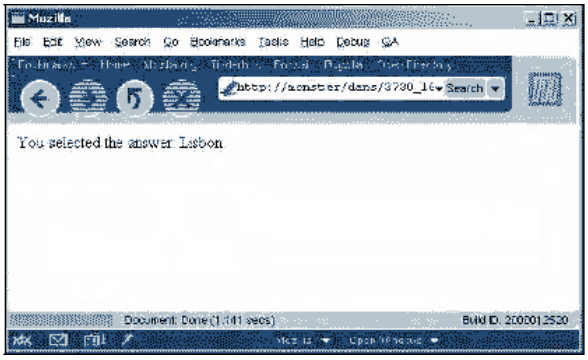
```
<HTML>
<HEAD></HEAD>
<BODY>
<?php
echo "You selected the answer: $Question1";
?>
</BODY>
</HTML>
```

4. Išsaugokite kaip `radio.php`.

5. Atidarykite `radio.html` savo naršyklėje ir pasirinkite atsakymą:



6. Paspauskite Submit Query ir pamatysite savo pasirinkimo rezultatus:



Kaip tai veikia

Mes vėl pasinaudojome `GET` metodu, tad rezultatai matomi URL eilutėje. Pirmasis puslapis, `radio.html`, padaro tris pasirinkimo mygtukus. Jie visi turi tą patį vardą, `Question1`, bet su trimis skirtingomis reikšmėmis, atspindinčiomis tris skirtingus atsakymus:

```
<INPUT NAME="Question1" TYPE="Radio" VALUE="Porto">
Porto
<BR>
<INPUT NAME="Question1" TYPE="Radio" VALUE="Lisbon">
Lisbon
<BR>
<INPUT NAME="Question1" TYPE="Radio" VALUE="Madrid">
Madrid
```

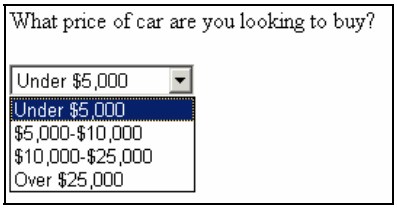
Tada mūsų antrasis puslapis, `radio.php`, pradeda dirbti jame esantis PHP kodas ir mums tereikia parodyti vieno kintamojo reikšmę, nes tegali būti tik vienas atsakymas į mūsų klausimą:

```
<?php
echo "You selected the answer: $Question1";
?>
```

Liko tik 2 formos elementai kuriuos mes turime apžvelgti, tad nieko nelaukdami tai padarykime.

Sąrašo langeliai (List Boxes)

Sąrašo langeliai arba išsiskleidžiantys sąrašai paprastai pateikia keletą variantų. HTML jie sukuriami su `<SELECT>` ir `<OPTION>` tag'ais. Iš esmės jie suteikia tokias pačias funkcijas kaip ir pasirinkimo mygtukai, paprastai suteikdami galimybę pasirinkti tik vieną dalyką iš sąrašo. Pavyzdžiui:

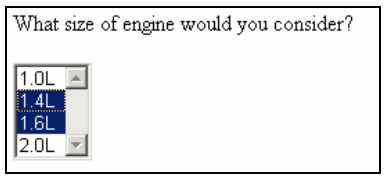


`<SELECT>` tag'as, kuris sukuria sąrašą turi keletą `<OPTION>` tag'ų. `<OPTION>` tag'ai savyje turi kiekvieną sąrašo elementą.

```
<SELECT NAME="Price">
    <OPTION>Under $5,000</OPTION>
    <OPTION>$5,000-$10,000</OPTION>
    <OPTION>$10,000-$25,000</OPTION>
    <OPTION>Over $25,000</OPTION>
```

```
</SELECT>
```

Tačiau galima vartotojui pasirinkti ir keletą sąrašo elementų vienu metu:



Jūs galite liesti keleto elementų pasirinkimą nustatydami `MULTIPLE` atributą `<SELECT>` tag'e. Tai priverčia PHP sukti galvą dėl dviejų dalykų. Bet geriau atlikime pavyzdį su abiems šiais elementais ir viską patys pamatysime. Mes paklausime vartotojo apie mašinos kainą ir jos variklio tūrį. Pirmasis klausimas leis pasirinkti tik vieną atsakymą; antrasis leis pasirinkti keletą (tai daroma paspaudus *Shift* klavišą).

Išbandykite – sąrašo langelių naudojimas

1. Atidarykite savo redaktorių ir parašykite sekantį kodą:

```
<HTML>
<HEAD></HEAD>
<BODY>
<FORM METHOD=GET ACTION="listbox.php">
What price of car are you looking to buy?
<BR>
<BR>
<SELECT NAME="Price">
<OPTION>Under $5,000</OPTION>
<OPTION>$5,000-$10,000</OPTION>
<OPTION>$10,000-$25,000</OPTION>
<OPTION>Over $25,000</OPTION>
</SELECT>
<BR>
<BR>
What size of engine would you consider?
<BR>
<BR>
<SELECT NAME="EngineSize[]" MULTIPLE>
<OPTION>1.0L</OPTION>
<OPTION>1.4L</OPTION>
<OPTION>1.6L</OPTION>
<OPTION>2.0L</OPTION>
</SELECT>
<BR>
<BR>
<INPUT TYPE=SUBMIT>
</FORM>
</BODY>
</HTML>
```

2. Išsaugokite kaip `listbox.html`.

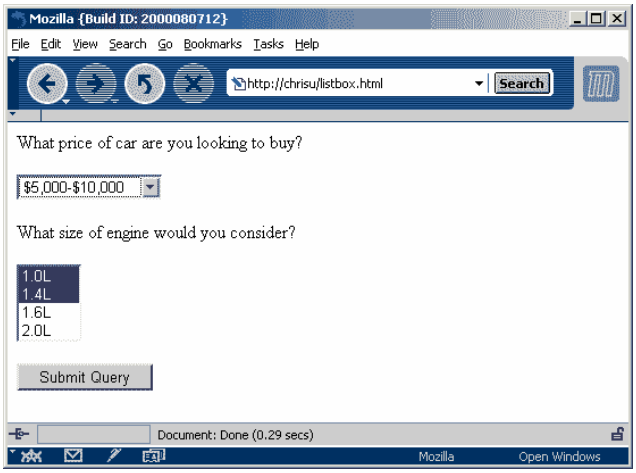
3. Uždarykite šitą failą ir atsidarykite naują:

```
<HTML>
<HEAD></HEAD>
<BODY>
<?php
echo "Price Range: $Price";
echo "<BR>Engine Size(s): $EngineSize[0]";
echo "$EngineSize[1]";
echo "$EngineSize[2]";
echo "$EngineSize[3]";
?>
```

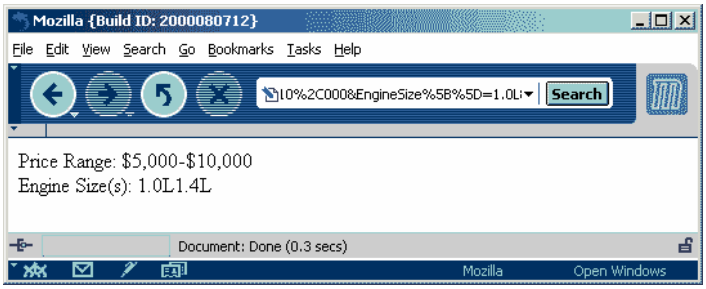
```
</BODY>
</HTML>
```

4. Išsaugokite kaip `listbox.php`.

5. Atidarykite `listbox.html` savo naršyklėje ir pasirinkite vieną reikšmę iš viršutinio sąrašo ir keletą iš apatinio:



6. Paspauskite ant Submit Query:



Kaip tai veikia

Padalinkime mūsų paaiškinimą kiekvieno sąrašo veikimo principų nagrinėjimą atskirai. Mūsų pirmame puslapyje `listbox.html`, mes sukūrėme sąrašo pasirinkimą su keturiom galimybėm, galima buvo pasirinkti tik

```
<SELECT NAME="Price">
    <OPTION>Under $5,000</OPTION>
    <OPTION>$5,000-$10,000</OPTION>
    <OPTION>$10,000-$25,000</OPTION>
    <OPTION>Over $25,000</OPTION>
</SELECT>
```

Antrajame puslapyje, `listbox.php`, mes nusiuntėme šį pasirinkimą į kintamąjį `$Price`.

```
<?php
echo "Price Range: $Price";
echo "<BR>Engine Size(s): $EngineSize[0]";
...
```

Čia nieko ypatingo nėra ir visa tai jums jau turėtu būti pažystama.
Mūsų antrasis pasirinkimo sąrašas kiek skiriasi nuo mūsų jau matytų:

```
<SELECT NAME="EngineSize[]" MULTIPLE>
<OPTION>1.0L</OPTION>
<OPTION>1.4L</OPTION>
<OPTION>1.6L</OPTION>
<OPTION>2.0L</OPTION>
</SELECT>
```

Na viskas irgi panašu išskyrus viršutinę eilutę. Viršutinė eilutė nustato atributą `NAME` kaip `EngineSize[]`. Tokį kintamąjį PHP skaitys kaip **masyvą** (array), čia vienas kintamasis savyje gali turėti daugybę kitų, visų jų vienas vardas, skiriasi tik pozicijos eilėje numeris – [1], [2], [3], ... Dabar tikiuosi jums kiek aiškiau.

```
echo "Price Range: $Price";
echo "<BR>Engine Size(s): $EngineSize[0]";
echo "$EngineSize[1]";
echo "$EngineSize[2]";
echo "$EngineSize[3]";
```

Mes sakėme, kad PHP kurdama masyvą, sukuria naują kintamąjį su tokiu pačiu vardu, bei indeksu. Mes turime keturis elementus sąrašą, tad bus keturių elementų masyvas. Juos reikia parodyti – žinoma jei jie pasirinkti. Masyvo indeksas visada prasideda nuo 0, `$EngineSize[0]` nurodo į pirmąjį pasirinkimą sąraše. Jie turės šią reikšmę tik tuo atveju, jei jie mes ją pasirinksime, kitu atveju pirmuoju numeriu eis bet kuris mūsų pasirinktas elementas iš sąrašo.

Šiuo atveju mes tikrai jį pasirinkome, tad `$EngineSize[0]` ištikro turi reikšmę 1.0L. Tas pats vyksta su `$EngineSize[1]` kuris turi antrąją reikšmę. `$EngineSize[2]` ir `$EngineSize[3]` neturi jokios reikšmės nes mes nepasirinkome daugiau reikšmių.

Paslėpti formos laukai

Yra atvėju kai jūs norite paimti informaciją esančią web puslapyje ir perduoti ją kitam puslapiui be vartotojo įsikišimo. Yra dar vienas `<INPUT>` tipas kuris leidžia įdėti informacija (ir ją siųsti) į tekstinį laukelį, tačiau vartotojui jis lieka nematomas.

Šitas tipas žinomas kaip **paslėpti formos laukai** (hidden form field).

Paslėpti formos laukai veikia kiek kitaip nei mūsų prieš tai aptarti formos elementai. Jie tikriausiai yra naudingiausi PHP puslapiuose kurie turi formas, nes jūs galite naudoti juos siųsdami kintamųjų reikšmes. Tipinis paslėptas laukelis atrodo taip:

```
<INPUT TYPE=HIDDEN NAME=Hidden1 VALUE="Secret Message">
```

Mes negalime parodyti jo paveiksliuko, nes šitas elementas nepasirodo ekrane. Bet kokia forma kuri buvo pasiųsta turės kintamąjį `$Hidden1` kurio reikšmė bus "Secret Message". Norint pasinaudoti paslėptais laukais PHP puslayje jūs galite parašyti visą HTML formą per `echo()` komandą – tokiu būdu jūs galite siųsti PHP kintamųjų reikšmes per HTML elementus:

```
<?php
$Message1="This message is invisible";
echo "<FORM>";
echo "<INPUT TYPE=HIDDEN NAME=Hidden2 VALUE='$Message1'>";
echo "<INPUT TYPE=SUBMIT>";
echo "</FORM>";
?>
```

Čia pateikta ištisa HTML forma parašyta PHP komandų viduje ir leidžianti mums sukurti kintamąjį `$Hidden2` ir suteikti jam `$Message1` reikšmę.

Padarykime pavyzdį kuris tari `<SELECT>` sąrašo laukelį ir pateikia vartotojo pasirinkimą bei keletą kitų opcijų sekančiame puslapyje. Mes taip pat visą šią formą parašysime įterptą PHP `echo()` komandos rėmus.

Išbandykite – Paslėptų laukelių naudojimas formose

1. Atidarykite savo redaktorių ir parašykite sekantį kodą:

```
<HTML>
<HEAD></HEAD>
<BODY>
<?php
$Message1="Bugs Bunny";
$Message2="Homer Simpson";
$Message3="Ren & Stimpy";
echo "<FORM METHOD=GET ACTION='hidden2.php'>";
echo "Which of the following would win in a shootout?";
echo "<SELECT NAME='ListBox'>";
echo "<OPTION>$Message1</OPTION>";
echo "<OPTION>$Message2</OPTION>";
echo "<OPTION>$Message3</OPTION>";
echo "</SELECT><BR><BR>";
echo "<INPUT TYPE=HIDDEN NAME=Hidden1 VALUE='$Message1'>";
echo "<INPUT TYPE=HIDDEN NAME=Hidden2 VALUE='$Message2'>";
echo "<INPUT TYPE=HIDDEN NAME=Hidden3 VALUE='$Message3'>";
echo "<INPUT TYPE=SUBMIT>";
echo "</FORM>";
?>
</BODY>
</HTML>
```

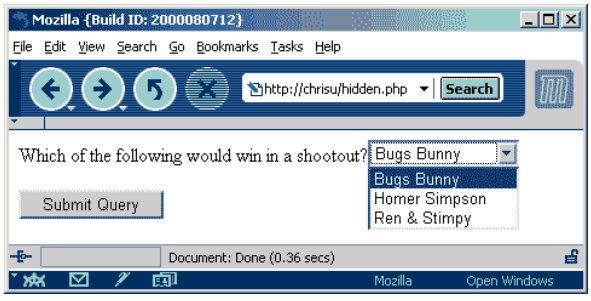
2. Išsaugokite kaip `hidden.php`.

3. Uždarykite šitą failą ir atsidarykite naują:

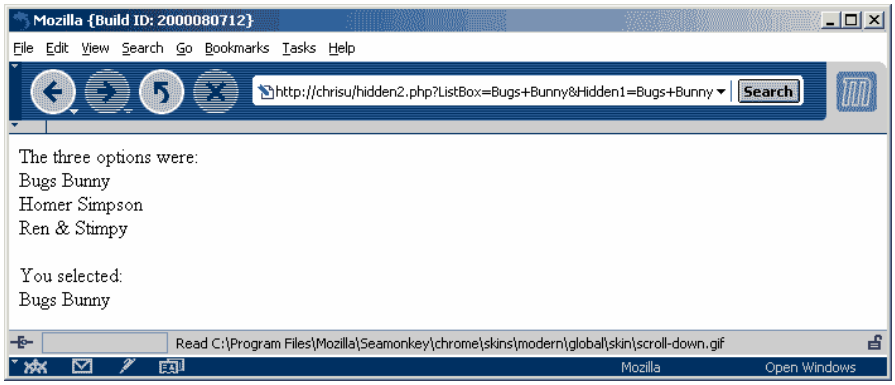
```
<HTML>
<HEAD></HEAD>
<BODY>
<?php
echo "The three options were:<BR>";
echo "$Hidden1<BR>";
echo "$Hidden2<BR>";
echo "$Hidden3<BR>";
echo "<BR>You selected:<BR>";
echo "$ListBox";
?>
</BODY>
</HTML>
```

4. Išsaugokite jį kaip `hidden2.php`.

5. Uždarykite šitą failą ir atsidarykite `hidden.php` su naršykle, tada pasirinkite:



6. Paspauskite ant Submit Query:



Kaip tai veikia

Kai jūs išmoksite kurti HTML formas `echo()` komandos viduje, vietoj paprastos HTML formos, jūs įsitikinsite, kad tai labai paprasta. Mes pradėjome sukurdami tris kintamuosius kurie suformuos pasirenkamo sąrašo `<SELECT>` elementus:

```
$Message1="Bugs Bunny";
$Message2="Homer Simpson";
$Message3="Ren & Stimpy";
```

Jie atitinkamai yra `$Message1`, `$Message2`, ir `$Message3`. Po to mes sukurėme HTML formą naudodamiesi `echo()` komandą. Visiškai nieko skirtingo muo paprastos formos, tiktai tai, kad mes turėjome naudoti ne dvigubas kabutes kaip HTML, o viengubas. Pirmoji linija paprasčiausiai nurodo siųsti informaciją į `hidden2.php` per `GET` metodą:

```
echo "<FORM METHOD=GET ACTION='hidden2.php'>";
```

Mes parodome šiek tiek paaiškinamojo teksto ir pradedame `<SELECT>` sąrašą:

```
echo "Which of the following would win in a shootout?";
echo "<SELECT NAME='ListBox'>";
```

Mes parašome tris galimus atsakymo variantus, kintamųjų `$Message1`, `$Message2`, ir
`$Message3` turinį atitinkamai.

```
echo "<OPTION>$Message1</OPTION>";
echo "<OPTION>$Message2</OPTION>";
echo "<OPTION>$Message3</OPTION>";
```

Tada mes uždarome `<SELECT>` sąrašą ir parašome keletą perkėlimų į kitą eilutę:

```
echo "</SELECT><BR><BR>";
```

Dabar mes paimame tris kintamuosius kuriuos mes jau naudojome ir per paslėptus laukelius pasiunčiame juos į mūsų formą:

```
echo "<INPUT TYPE=HIDDEN NAME=Hidden1 VALUE='$Message1'>";
echo "<INPUT TYPE=HIDDEN NAME=Hidden2 VALUE='$Message2'>";
echo "<INPUT TYPE=HIDDEN NAME=Hidden3 VALUE='$Message3'>";
```

Šie trys kintamieji pasidarys reikšmėmis kitų trijų kintamųjų - `$Hidden1`, `$Hidden2`, ir `$Hidden3` atitinkamai. Tada mes galime parašyti Submit mygtuką, ir uždaryti formą:

```
echo "<INPUT TYPE=SUBMIT>";
echo "</FORM>";
```

Antrasis PHP puslapis parasčiausiai parodo pirmojo sukurtus kintamuosius ir jų reikšmes. Visų pirma mes parodome tris paslėptus laukelius:

```
echo "The three options were:<BR>";
echo "$Hidden1<BR>";
echo "$Hidden2<BR>";
echo "$Hidden3<BR>";
```

Tai yra labai naudinga nes paprastai visos sąrašo laukelio reikšmės nėra persiunčiamos. Tiktai vartotojo pasirinktos reikšmės bus perduotos į kitą PHP puslapį. Tačiau kartais jums reikia turėti visas sąrašo reikšmes pasiekiamas PHP puslapyje. Tai yra vienas efektyviausių metodų persiųsti šitokio tipo informaciją.
Paskutinė linija parodo vartotojo pasirinkimą.

```
echo "<BR>You selected:<BR>";
echo "$ListBox";
```

Mes ir toliau knygoje naudosime paslėptus laukelius atlikdami tokio tipo užduotis.

Slaptažodžiai

Slaptažodžiai yra paprasčiausi tekstiniai laukeliai, kurie pakeičią įvedamą tekstą į žvaigždutes. Jie saugo ir perduodą informaciją tokiu pačiu būdu kaip ir tekstiniai laukeliai.

```
Koks jūsų slaptažodis?
<INPUT NAME="Password" TYPE="Password">
```

Mes neatliksime su jais praktinio pavyzdžio, nes nėra skirtumo tarp `TEXT` ir `PASSWORD` tipo laukų. Jei norite pamatyti kaip jis dirba, grįžkite atgal ir su prieš tai buvusiame `text.html` pakeiskite tipą į `PASSWORD`. Tačiau jei jūs sugalvosite siųsti šią informaciją naudodami GET, pastebėkite, kad ši informacija nėra šifruojama ir ją gali matyti visi.
Tai nereiškia, kad `POST` yra saugus duomenų siuntimo būdas, paprasčiausiai informacija nebus matoma taip akivaizdžiai. Jei norite saugumo jums reikia naudoti kažką panašaus į SSL (Secure Sockets Layer).

Submit ir Reset mygtukai

Mes jau naudojome Submit mygtukus pavyzdžiuose šiame skyriuje, tad nerodysime kaip jie veikia. Tačiau yra keletas dalykų kuriuos reiktu pažymėti. Kas atsitiktu jei jums reiktu daugiau nei vieno Submit tipo mygtuko puslapyje? Tokiu atveju jums reiktu nurodyti mygtuko atributus `NAME` ir `VALUE`. Pavyzdžiui:

```
  <INPUT VALUE="Button 1 pressed" TYPE="SUBMIT"
NAME="Submit1">
  <INPUT VALUE="Button 2 pressed" TYPE="SUBMIT"
NAME="Submit2">
```

Tai, kaip galite tikėtis sukuria kintamuosius PHP su kuriais jis gali dirbti. Jei nagrinėti viršuje parašyta kodą, jis sukurs tik vieną kintamąjį, priklausomai nuo to koks mygtukas buvo paspaustas. Jei paspausite Submit1 tada yra sukuriamas kintamasis pavadintas `$Submit1`. Jei paspausite Button 2, tada sukuriamas `$Submit2`. `$Submit1` reikšmė yra "Button 1 pressed", tuo tarpu `$Submit2` turinys yra "Button 2 pressed". Mes negalime daryti nieko naudingo kol kas, tad nerodysime pavyzdžio. Ketvirtame skyriuje mes susipažinsime su papildomom programavimo žiniom ir panaudosime šį dvigubo mygtuko privalumą.

Submit tipo mygtukas neleidžia greitai ištrinti informacijos jei ji buvo įvesta neteisingai. Nors jūs negalite atšaukti informacijos pasiųstos per Submit mygtuką, Reset mygtukas dažnai būna naudingas kai reikia sugražinti visus formos elementus į jų pradinę būseną.

```
<INPUT TYPE="Reset">
```

Dabar kai susipažinome su visais formos elementais kuriuos mes naudosime šioje knygoje, mes sudėsime juos kartu ir padarysime vieną didelį pavyzdį.

Reikšmių gautų iš formos naudojimas PHP skripte

Mes pademonstravome visu formos elementus ir kaip PHP su jais dirba, bet mes iki šiol nepadarėme nieko išskyrus gautų reikšmių rodymą kitame puslapyje. Reikia pripažinti, kad be visų galimybių kurias mes nagrinėsime kitame skyriuje tai padaryti sunku. Tačiau mes jau išmokome atlikti matematines ir tekstines operacijas prieš tai buvusiame skyriuje, tad sujungia šias žinias su dabar gauta informacija galime parašyti praktinį pavyzdį.

Paskutiniame šio skyriaus pavyzdyje mes sukursime paskolos programą (formą), kuri klausia apie pinigų kiekį kurį asmuo nori pasiskolinti, ir suskaičiuosime kiek mūsų sugalvotas bankas NAMLLU gali pasiūlyti klientui atsižvelgdami į jo amžių ir uždarbį. Skaičiavimo gale mes pateiksime paprasčiausia Taip arba Ne. Nors mūsų formulė gali pasirodyti didelė, ji yra paprasta ir nepagrysta realiom formulėm kurios yra žymiai sudėtingesnės

Paskolos suma mūsų programoje yra skaičiuojama pagal tris faktorius:

- Visų pirma apskaičiuosime metinės algos penktadalį, taip gaudami algos dydį kurį galima paskirti skolos išmokėjimui.

- Po to padalinsime prašančiojo amžių iš 10, o gautą atsakymą suapvalinsime iki artimiausio sveiko skaičiaus.

- Tada pasinaudoja antruoju skaičiavimu apskaičiuosime leistiną amžių.

- Pati formulė paima pirmąjį skaičiavimą ir padauginsime iš antrosios, taip gaudami mūsų paskutinį atsakymą – leistinos paskolos dydį.

Antrasis punktas reikalingas tam, kad bet kas turintis mažiau nei 20 metų būtu išbrauktas iš galimų paskolos ėmėjų, nes formulė visada pateiks 0. Pažiūrėkime pavyzdį ir viskas taps aiškiau.

```
First figure * (19/10 – (19 Modulus 10) /10))-1
```

Atsiminkite, kad modulud operatorius naudojamas gauti dalybos liekaną. Tada mūsų skaičiavimas bus panaudotas taip:

```
First figure * (1.9 – 0.9) -1
```

Kuris galiausiai pateikia 0

```
First figure * 0
```

Taigi jei jūs pateiksite amžių jaunesnį nei 20, formulė visada pateiks 0 kaip atsakymą. Nesvarbu kokie kiti skaičiai įvedami, daugyba iš nulio visada duos atsakymą 0. Kai padauginame mūsų pirmąjį punktą iš antrojo, gauname leistinos paskolos sumą – jei žmogus nori mažiau mes sakome “Taip jūs galite skolintis”, jei leistina suma mažesnė nes atsisakome išduoti paskolą.

Dar kartą mus reikės dviejų puslapių. Pirmasis bus paprasčiausia forma kuriame mes sužinosime kliento vardą, pavardę, amžių, adresą, algą, bei norimos paskolos dydį.

Mes naudosime beveik visus formos elementus aptartus šiame skyriuje. Antrame, PHP tipo, puslapyje mes atliksime savo apskaičiavimus ir pateiksime verdiktą.

Išbandykite – paskolos prašymo forma

1. Atidarykite savo failų redagavimo programą ir parašykite:

```
<HTML>
<HEAD></HEAD>
<BODY>
<B>Namllu Credit Bank Loan Application Form</B>
<FORM METHOD=POST ACTION="loan.php">
First Name:
<INPUT NAME="FirstName" TYPE="Text">
Last Name:
<INPUT NAME="LastName" TYPE="Text">
Age:
<INPUT NAME="Age" TYPE="Text"SIZE="3">
<BR>
<BR>
Address:
<TEXTAREA NAME="Address" ROWS=4 COLS=40>
</TEXTAREA>
<BR>
<BR>
What is your current salary?
<SELECT NAME="Salary">
<OPTION VALUE=0>Under $10000</OPTION>
<OPTION VALUE=10000>$10,000 to $25,000</OPTION>
<OPTION VALUE=25000>$25,000 to $50,000</OPTION>
<OPTION VALUE=50000>Over $50,000</OPTION>
</SELECT>
<BR>
<BR>
How much do you want to borrow?<BR><BR>
<INPUT NAME="Loan" TYPE="Radio" VALUE=1000>Our $1,000
package at 8.0% interest
<BR>
<INPUT NAME="Loan" TYPE="Radio" VALUE=5000>Our $5,000
package at 11.5% interest
<BR>
<INPUT NAME="Loan" TYPE="Radio" VALUE=10000>Our $10,000
package at 15.0% interest
<BR>
<BR>
<INPUT TYPE=SUBMIT VALUE="Click here to Submit
application">
<INPUT TYPE=RESET VALUE="Reset application form">
</FORM>
</BODY>
</HTML>
```

2. Išsaugokite kaip `loan.html`

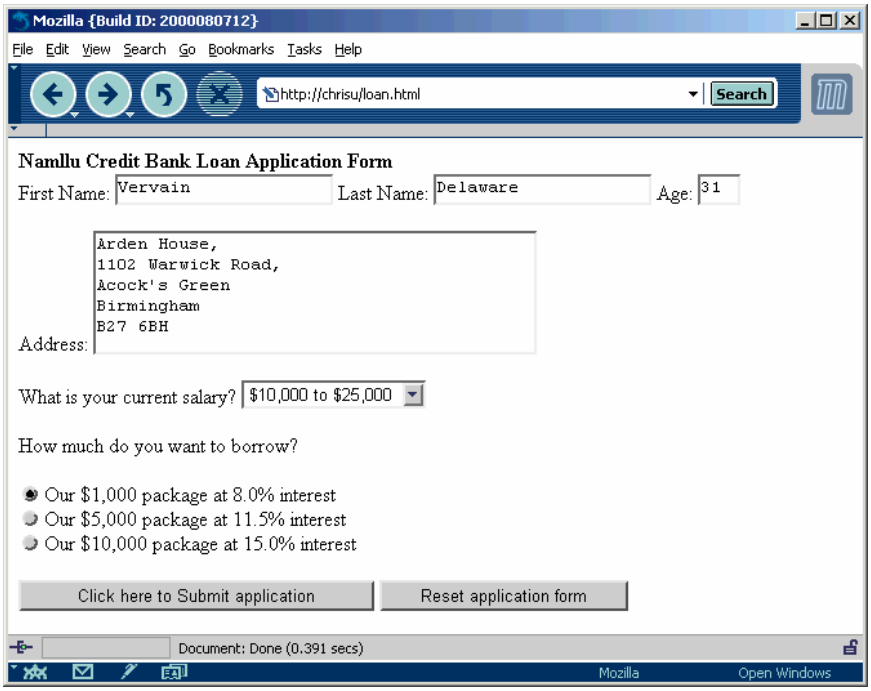
3. Uždarykite pirmąjį failą ir pradėkite naują:

```
<HTML>
<HEAD></HEAD>
<BODY>
<B>Namllu Credit Bank Loan Application Form</B>
<BR>
<BR>
<?
$SalaryAllowance = $Salary/5;
$AgeAllowance = ($Age/10 - ($Age%10)/10)-1;
$LoanAllowance = $SalaryAllowance * $AgeAllowance;
echo "Loan wanted:$Loan<BR>";
echo "Loan amount we will allow:$LoanAllowance<BR><BR>";
```

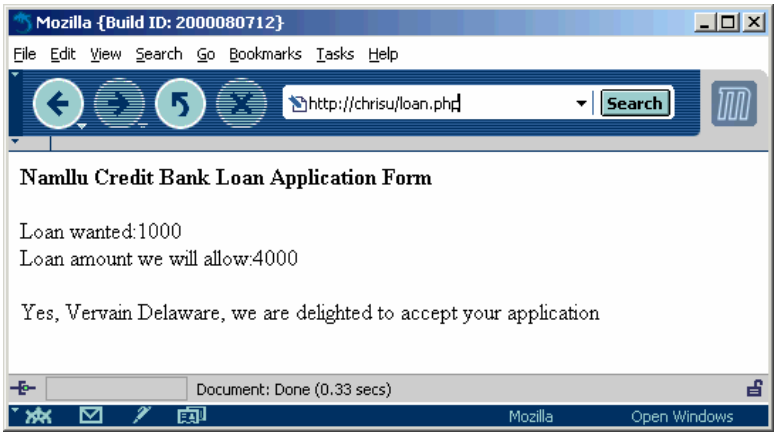
```
  if ($Loan <= $LoanAllowance) echo "Yes, $FirstName
$LastName, we are delighted to
  accept your application";
  if ($Loan > $LoanAllowance) echo "Sorry, $FirstName
$LastName, we cannot accept
  your application at this time";
  ?>
  </BODY>
  </HTML>
```

4. Išsaugokite kaip `loan.php`

5. Atidarykite `loan.html` savo interneto naršyklėje ir įveskite kokius nors duomenis:



6. Paspauskite Submit application mygtuką ir turėtumėte gauti panašų atsakymą.



Kaip tai veikia

Jūs tikrai užsitarnavote pertrauka po to kai mes išnagrinėsime šį pavyzdį. Pirmasis HTML tipo puslapis yra gana paprastas, jo elementus mes išnagrinėjome anksčiau. Mes iš viso sudėjome 8 formos elementus į `loan.html`. Pirmieji trys yra:

```
First Name:
<INPUT NAME="FirstName" TYPE="Text">
Last Name:
<INPUT NAME="LastName" TYPE="Text">
Age:
<INPUT NAME="Age" TYPE="Text"SIZE="3">
```

Visi šie elementai yra tekstiniai laukeliai, naudojami gauti vardą, pavardę ir amžių. Dabar jūs jau turėtumėte suprasti, kad visi šie elementai sukurs kintamuosius `$FirstName`, `$LastName`, ir `$Age` mūsų PHP puslapyje.

Adresas gaunamas per `<TEXTAREA>` tipo elementą:

```
<TEXTAREA NAME="Address" ROWS=4 COLS=40>
</TEXTAREA>
```

Savo ruoštu tai sukuria PHP kintamąjį `$Address`. Jūs tikriausiai jau pastebėjote, kad mes nenaudojame visų gautų duomenų, tačiau vėlesniuose skyriuose mes grįšime prie šio pavyzdžio ir naudosime ir kitus elementus.

Sekantis elementas yra sąrašo laukelis, kuriame mes pateikiame algos dydį:

```
<SELECT NAME="Salary">
<OPTION VALUE=0>Under $10000</OPTION>
<OPTION VALUE=10000>$10,000 to $25,000</OPTION>
<OPTION VALUE=25000>$25,000 to $50,000</OPTION>
<OPTION VALUE=50000>Over $50,000</OPTION>
</SELECT>
```

Mes iš tikro negalime išsaugoti skaičių intervalo, tad mes vietoj to paimame mažiausią intervalo skaičių ir priskiriame šią reikšmę prie kintamojo. Sąrašo laukelis sukuria tiktai vieną kintamąjį `$Salary`, ir suteikia jam reikšmę priklausomai nuo pasirinktos sumos. Jei nebuvo pasirinkta nieko sąrašo laukelis nepateiks jokios reikšmės. Atkreipkite dėmesį, kad pirmam pasirinkimui mes suteikėme reikšmę 0, tai užtikrins, kad visi gaunantys mažiau nei $10,000 automatiškai bus atmesti.

Kitas formos elementas yra trys susyją pasirinkimo mygtukai:

```
How much do you want to borrow?<BR><BR>
<INPUT NAME="Loan" TYPE="Radio" VALUE=1000>Our $1,000
package at 8.0% interest
<BR>
<INPUT NAME="Loan" TYPE="Radio" VALUE=5000>Our $5,000
package at 11.5% interest
<BR>
<INPUT NAME="Loan" TYPE="Radio" VALUE=10000>Our $10,000
package at 15.0% interest
<BR>
```

Jie visi turi tą patį vardą, nes kintamasis turi turėti tik vieną reikšmę, priklausomai nuo to ką pasirinko vartotojas. Ši grupė sukuria tik vieną PHP kintamąjį - `$Loan`.

Paskutiniai elementai yra Submit ir Reset mygtukai:

```
<INPUT TYPE=SUBMIT VALUE="Click here to Submit
application">
<INPUT TYPE=RESET VALUE="Reset application form">
```

Submit mygtukas inicijuoja `ACTION` atributą, kuris parašytas formos pradžioje, ir taip forma žino kur siųsti savo duomenis:

```
<FORM METHOD=POST ACTION="loan.php">
```

Paskolos apskaičiavimo vykdymas

Mes aptarėme informacijos gavimą ir jos persiuntimą; antrasis mūsų failas `loan.php` paima šias reikšmes ir atlieka paprastus skaičiavimus taip išsiaiškindamas ar pritarti ar atmesti paskolos gavėjo prašymą. Pažiūrėkime kaip jums sekėsi suprasti kas gi vyksta. Pirmoji linija sukuria naują kintamąjį $Salary Allowance, kuriam priskiriama algos penktadalio reikšmė:

```
$SalaryAllowance = $Salary/5;
```

Antroji linija apskaičiuoja kiek sudėtingesnį leistiną amžių. Šia formule mes norime gauti sveiką skaičių padalinę prašytojo amžių iš 10, jei po dalybos veiksmo lieka kokia nors reikšmė po kablelio, mes ją su apvaliname iki artimiausio sveiko skaičiaus.

Programuotojai kitomis kalbomis pastebės, kad tai ką mes darome yra naudojame DIV funkciją. Deja PHP neturi tokios standartinės funkcijos

Apskaičiuodami likutį mes panaudojame modulus operatorių. Mūsų paskutinė linija duos rezultatą 0, jei amžius bus tarp 0 ir 19, 1 jei amžius bus tarp 20-29, 2 jei įvesta bus 30-39 ir .t.t. Šio skaičiavimo rezultatai yra išsaugomi naujame kintamajame `$AgeAllowance`:

```
$AgeAllowance = ($Age/10 - ($Age%10)/10)-1;
```

Gerai, kad sekanti kodo linija daug paprastesnė. Ji paima mūsų ką tik apskaičiuotus kintamuosius ir padaugina juos. Gauti duomenys vėl gi išsaugomi naujame kintamajame `$LoanAllowance`, kuris ir yra mūsų galutinis atsakymas kokio dydžio maksimali paskola gali būti.

```
$LoanAllowance = $SalaryAllowance * $AgeAllowance;
```

Sekančios dvi linijos yra paprasčiausiai `echo()` parodančios kiek vartotojas norėjo gauti ir kiek mes galime jam duoti paskolos.

```
echo "Loan wanted:$Loan<BR>";
echo "Loan amount we will allow:$LoanAllowance<BR><BR>";
```

Jei iki šiol supratote kaip viskas vyksta, labai gerai. Bet sekančiose dviejuose linijose mes truputėli pasukčiavome ir pristatėme naują galimybę <= (mažiau arba lygu operatorių). Tai leidžia mus priimti sprendimą, pagal gautus duomenis. Šis operatorius apskaičiuoja ar norima paskolos suma yra mažesnė ar lygi sumai kurią gali išduoti bankas. Jei mažesnė mes parodome pranešimą, kad paskola gali būti išduota. Mes išsamiai išnagrinėsime šią ir kitas sąlygos struktūras sekančiame skyriuje, tad nepanikuokite jei dabar dar ne viską supratote.

Mes taip pat personalizavome šį pranešimą pateikdami formoje įvestą vardą ir pavardę.

```
  if ($Loan <= $LoanAllowance) echo "Yes, $FirstName
$LastName, we are delighted to accept your application";
```

Paskutinė kodo eilutė yra priešingybė prieš tai buvusiai. Jei paskolos dydis viršyja banko nustatytą, mes patekiame pranešimą, kad paskola negalima.

```
  if ($Loan > $LoanAllowance) echo "Sorry, $FirstName
$LastName, we cannot accept
  your application at this time";
```

Tai tiek mūsų programos.

Galimas formos pagerinimas

Negalima pasakyti, kad mūsų forma ideali; jei gerai pasistengti galima ją pergudrauti arba priversti rodyti nelogiškas reikšmes. Taip atsitinka nes mes nenaudojome jokios įvestos informacijos patikrinimo. Kas gi neleidžia vartotojui

įvesti 965 kaip savo amžių? Mes žinome, kad tai ne tiesa, bet negalime tos sustabdyti. Sekančiame skyriuje mes apžvelgsime į būdus kaip išvengti panašių situacijų ir kaip tikrinti duomenis.

Apibendrinimas

Šis skyrius galėjo pasirodyti kiek pedantiškas, bet koncepcijos kurias mes nagrinėjome yra vienos kertinių PHP pagrindų. Bet koks formos naudojimas reiškia, kad jūs turite gerai mokėti suvaldyti dešimtis ir daugiau elementų, priversdami juos dirbti sau. Tad mes stengėmės sudėlioti viską į savo vietas, kad sekančiuose skyriuose nebūtu problemų ir nereikėtu kartotis.

Mes pradėjome nagrinėdami `HTML <FORM>` tag'ą su visomis detalėmis ir sužinojome apie du galimus informacijos siuntimo metodus. Pirmasis yra `GET` metodas, kuris viešai perduoda informaciją prikabintą prie URL. Antrasis yra `POST` metodas, kuris siunčia formos informaciją paslėptą, HTML viduje, šis būdas diskretiškesnis, tačiau iš tikro ne ką ne saugesnis. Mes išsiaiškinome, kad `ACTION` atributas naudojamas nurodant į kokį puslapį siusti gautą informaciją.

Mūsų pagrindinių HTML formos elementų nagrinėjimas irgi davė naudos nes kaip tik per šiuos elementus informacija pasiekia PHP ir ji gali dirbti su duotais duomenimis.

Pačioje pabaigoje mes panaudojome beveik visus mums žinomus elementus dideliame pavyzdyje – rinkome informacija, atlikome matematinius veiksmus bei pateikėme rezultatus. Šiame pavyzdyje mes pristatėme naują sprendimų priėmimo koncepciją. Ši sprendimų priėmimo struktūra, priklausomai nuo sąlygų, leis elgtis vienaip ar kitaip. Tai ir bus kito skyriaus pagrindinė tema.

4

Sprendimų priėmimas

Prieš tai buvusiame skyriuje, mes išmokome siųsti formos informaciją ir gauti dinaminį atsakymą iš serverio, kurį mes vėliau pateikiame mūsų web puslapyje. Reikia pripažinti, mes vis dar suvaržyti dalykų visuma kuriuos mes galime atlikti. Paskutiniame pavyzdyje 3 skyriuje mes netgi truputi gudravome, panaudodami naują galimybę, kuri parodo arba ne kodo liniją priklausomai nuo sąlygų.

Mes gilinsimės į pačius PHP „vidurius“ sekančiuose trijuose skyriuose, pradedant procesais kuriuos naudodama PHP priima sprendimus. Be to ankščiau jūsų PHP kodas buvo atliekamas griežtai pagal eiliškumą – pirma linija, antra linija ir t.t. Sprendimų priėmimo galimybė leidžia jums nuspręsti ar vykdyti kažkokią kodo dalį, bei leis jums vykdyti palyginimus tarp kintamųjų ir jų reikšmių.

Tai taip pat leis jums susipažinti su kai kuriais loginiais (boolean) operatoriais – kurie gražina reikšmes "true" ar "false"). Mes galėjome juos pristatyti jau antrame skyriuje kartu su netematiniais ir eilutės tipo operatoriais, bet mes dar negalėjome pademonstruoti jų panaudojimo ir efektyvumo. Jūs įsitikinsite, kad galimybės išmoktos šiame skyriuje leis jums rašyti daug sudėtingesnius PHP skriptus, be to mes juos panaudosime pagerindami 3 skyriaus baigiamąjį pavyzdį.

Šiame skyriuje aptarsime šias temas pagal eilę

- Kaip šakojimai įtakoja kodo vykdymą
- Kasdieninis šakojimo pavyzdys.
- `If` elementas
- Palyginimo operatoriai
- Lygybės operatoriai
- Loginiai operatoriai
- `Switch` elementas
- Kaip patikrinti formos turinį

Sąlygos arba šakojimosi operatorius

Pačia paprasčiausia prasme, sąlygos kodas reiškia, kad „atlik vieną kodo eilutę“ arba „neatlik jos“, priklausomai nuo to ar atitinka nurodytas kriterijus. Kiek sudėtingesnis variantais gali reikšti, kad „atlik šią kodo eilutę“ arba „atlik šitą eilutę“, priklausomai nuo to ar atitinka nurodytas kriterijus. Jūs galite padidinti tai nurodydami atlikti visą kodo gabalą vienu atveju, arba kažkokį kitą kodą priešingu atveju. Kad geriau viską suprastumėte pateiksime mažą pavyzdėlį. Jei kažkokių apskaičiavimų rezultatas yra 1 bus vykdoma pirmoji eilutė, jei rezultatas 2 bus atliekama antroji eilutė, jei 3 tai trečioji ir t.t.

Šakojimosi pavyzdys iš kasdieninio gyvenimo

Prieš tai mes kalbėjome daugiau abstraktiškai, tad paremkime mūsų žinias atlikdami pavyzdį iš kasdieninio gyvenimo. Apsipirkimas yra tikriausiai kasdieniškiausias veiksmas apie kurį mes galime galvoti, bet jis tinka parodant sprendimų priėmimo pavyzdį kurį gali atlikti jūsų programa.

Įsivaizduokite jums reikia atlikti žemiau pateikta sąrašą:

a) padaryti puodelį arbatos
b) pasigaminti sumuštinį su sūriu
c) pamaitinti gyvūnus

Jums taip pat reikės eiti ir nusipirkti visus produktus, jei jūs jų neturite, tačiau sakykime jūs teturite 5 dolerius. Dabar mes visą tai pavaizduosime lietuvių kalbos terminais:

1. Patikrink šaldytuvą ar jame yra pieno, sūrio ir sviesto, jei nėra įrašyki į sąrašą.
2. Patikrink duoninę ar joje yra duonos, jei nėra įrašyk į sąrašą.
3. Patikrink bufetą ar jame yra arbatos jei nėra įrašyk į sąrašą.
4. Jei visi produktai yra norint atlikti pirmus tris punktus eikite prie 7‘ojo punkto.
5. Nueikite į supermarketą
6. Nusipirkite tiek produktų esančių sąraše kiek užteks nusipirkti už 5 dolerius.
7. Jie turite gyvūnų ėdalo, pamaitinkite gyvūnus. Jei neturite praleiskite 8 punktą.
8. Gyvūnai yra laimingi.
9. Jei turite pieno ir arbatos, jūs galite pasidaryti puodelį arbatos. Jei ne praleiskite 10 punktą.
10. Jūs nebesate ištroškęs.
11. Jei turite duonos, sviesto ir sūrio, jūs galite pasidaryti sumuštini su sūriu. Jei taip nėra, praleiskite 12 punktą.
12. Jūs nebe alkanas.
13. Pavargęs išsitieskite priešais televizorių!

Labai mažai tikėtina, kad jūs atliksite visu 13 programos žingsnių (nors tai ir nėra tikra kompiuterinė programas mes ją taip vadinsime, nes ji turi tokią pačia logiką). Priklausomai nuo ankstesnių žingsnių rezultatų jūs praleisite tam tikrus punktus, arba jei bus patenkinti pirmi 3 punktai bus praleista didžioji programos dalis. Taip pat yra įmanoma ir visai kitokia baigtis jei jūs neišnaudosite savo pinigų protingai. Tai tipiškas sprendimų priėmimas atliekamas PHP kalboje. Kai jūs rašote PHP programas, dalis kodo bus skirta specifinėms situacijoms, bet jei jos nekils tos kodo dalys bus praleistos kaip nereikalingos.

Jūs gana lengvai galite pertekti visą šį pavyzdį PHP programoje. Mes galėtumėme pavaizduoti kiekvieną mūsų apsipirkimo ekspedicijos eilutę kaip kodą. Jūs turėtumėte pradėti nuo pirmosios linijos kuriai pavaizduoti savo ruožtu reikės trijų kodo eilučių, nes mums reikės atskirų veiksmų priklausomai nuo sąlygų. Patikrinti šaldytuvą ar jame yra pieno, patikrinti šaldytuvą ar jame yra sūrio, patikrinti šaldytuvą ar jame yra sviesto. Norint pavaizduoti pirmąjį veiksmą mums tereikia vieno `if` operatoriaus kuria patikrintu sąlygą ir atliktu veiksmus jei rezultatas teigiamas:

```
  if ($SaldytuveNeraPieno) $PirkiniuSarasas =
$PirkiniuSarasas . "Pienas.";
```

Tariant kitas žodžiais, jei šaldytuve nėra pieno, mes pridedame pieną prie pirkinių sąrašo. Tą patį mes turėtumėme atlikti ir su kitais produktais. Tačiau vietoj pavertimo kiekvienos eilutės į PHP kodą, geriau judėkime toliau ir apžvelkite taisykles kurios valdo `if` operatorių.

If operatorius

Mes jau minėjome `if` operatorių ankstesniame skyriuje, tad vėl su juo susidūrę jūs jau turėtumėte numanyti kaip jis veikia.
Abstrakčiai paėmus jis veikia štai taip:

`if (sąlyga yra teisinga)` *atlikti kodo linija*

`If` operatorius atliks bet kokį kodą jei sąlyga yra patenkinta. Jei sąlyga nepatenkinta, kodas bus visiškai ignoruojamas, ir PHP jo visiškai neatliks. Programa paprasčiausiai pradės vykdyti sekančia kodo eilutę.

`if (`*oras yra lietingas*`)` *užsidedame skėti*
einame laukan

Antroji eilutė bus atliekama nežiūrint į nieką, tačiau mes skėti užsidėsime tik tuo atveju jei oras bus lietingas. Jei jums reikia atlikti visą kodo gabalą, jūs turite parašyti kodą atskirose eilutėse po sąlygos operatoriaus, tarp riestinių skliaustelių:

`if (`*sąlyga yra patenkinta*`)`
`{`
Atliekamas kodas tarp šių skliaustelių
`}`

Tad norin išplėsti mūsų lietsargio pavyzdį mes galėtume sakyti:

`if (`*oras yra lietingas*`)`
`{`
užsidedame skėti
užsidedame lietpaltį
`}`

einame laukan

Dar kartą „einame laukan“ bus visuomet vykdoma, tačiau skėti ir lietpaltį mes užsidėsime tik tuo atveju jei „oras yra lietingas“ atitinka tikrovę.

Dabar pažiūrėkime kaip jūs galite sukurti sąlygas remiantis kuriomis kodas bus atliekamas arba ne. Tiesa pasakius nėra jokios reikšmės kas yra tarp skliaustelių, `if` operatorius paprasčiausiai patikrina sąlygos atitikimą ir paleidžią kodą arba ne.

Loginės reikšmės

Prieš mums einant toliau, mes turime pristatyti jums **loginių** reikšmių sąvoką. Prieš tai buvę kintamieji galėjo turėti skaitines arba tekstines reikšmes, tačiau loginės reikšmės yra trečias kintamųjų tipas, galintis turėti vieną iš dviejų absoliučių reikšmių **true** ar **false**. Jūs galite priskirti bet kokiam kintamajam vieną iš šių reikšmių:

```
$kintamasis = true;
```

Tačiau jei jūs o to pavaizduosite reikšmę ekrane, jūs gausite skaitinę reikšmę:

```
1
```

Tas pats vyksta su `$kintamasis` reikšme `false`, bus pavaizduota:

```
0
```

Taigi jūs matote, loginės reikšmės turi abidvi reikšmes – skaitines ir raidines. Savaime tai nėra labai įdomu, tačiau kai jums reikės priimti sprendimus gautose situacijose, tame tarpe kai rezultatas bus loginis- taip arba ne, šis faktas bus labai svarbus.

Loginiai operatoriai.

Iki šiol mes smulkiai nagrinėjome matematines operacijas (kurių kaip matėte nėra labai daug), yra ir kitų operatorių grupė kalbą apie kuriuos mes atidėjome. Taip padaryta todėl, kad be šakojimosi struktūros šie operatoriai yra beverčiai. Paskutiniame skyriuje mes jau pristatėme kai kuriuos iš jų, kad galėtumėme atlikti sprendimą paskolos davimo formoje. Tiesa pasakius bet kada kai jums reikės sukurti sąlygą norint priimti bet kokį sprendimą, jūs turėsite naudotis šiais operatoriais. Mes padalinsime juos į keturias grupes ir pažiūrėsime į kiekvienos grupės operatorius pavyzdžiuose.

> ir < operatoriai

Jūs jau turėtumėte žinoti daugiau nei ir mažiau nei operatorius - tai fundamentalūs operatoriai netgi paprastoje matematikoje, bei labai svarbūs programavime. PHP kalboje mes galime juos panaudoti palygindami dvi konstantas, konstantą su kintamuoju ar du kintamuosius. Priklausomai nuo palyginimo rezultatų, tam tikra veiksmų seka gali būti atliekama. Su šiomis konstantomis rezultatas yra akivaizdus:

```
If (5 < 6) echo "Five is smaller than six";
```

Tačiau mes vis tiek turime plačiau pažvelgti kas gi vyksta. Sąlygos dalis `if` operatoriuje yra apgaubta skliausteliais. Ji gali apskaičiuoti ir gauti vieną iš dviejų loginių reikšmių, true ar false. Arba sąlyga yra patenkinta arba ne. Ji negali būti dalinai patenkinta, ir PHP negali pateikti reikšmės „Galbūt“ ar „Aš dar pagalvosiu“. Taigi šio kodo rezultatas yra „true“. `If` operatorius atliks veiksmus tik tada kai sąlyga yra true – ji yra patenkinta.

Mūsų pavyzdys nėra labai naudingas – jūs ir taip žinote, kad 5 mažiau nei 6. Tačiau, jei mes lygintume kintamojo reikšmę su skaičiumi, viskas priklausytu nuo kintamojo reikšmės:

```
If ($Kintamasis < 6) echo ("Kintamasis yra mažesnis nei
šeši");
```

Arba mes galime palyginti du kintamuosius:

```
If ($Kintamasis1 < $Kintamasis2) echo ("Mūsų kintamasis
nr.1 mažesnis nei kintamasis nr.2");
```

Ir žinoma šio palyginimo rezultatus mes galime naudoti, ne tik parodant pranešimą bet ir imantis tam tikrų veiksmų:

```
If ($Kintamasis1 < $ Kintamasis2)
{
echo ("Mūsų kintamasis nr.1 yra per mažas");
$Kintamasis1 = $Kintamasis1+1;
}
```

Gerai, atlikime nedidelį pavyzdį kuriame mūsų PHP programa „sugalvotu“ skaičių tarp 1 ir 10, o mums tektu jį atspėti. Tam, kad PHP „sugalvotu“ skaičių mes naudosime PHP atsitiktinių skaičių generavimo funkciją `rand`. Po to kai atliksime pavyzdį mes paaiškinsime funkcijos veikimo principą.

Išbandykite – Palyginimo operatorių naudojimas

1. Atidarykite savo teksto redagavimo programą ir parašykite:

```
<HTML>
<HEAD></HEAD>
<BODY>
<FORM METHOD=GET ACTION="guessgame.php">
What number between 1 and 10 am I thinking of?
<INPUT NAME="Guess" TYPE="Text">
<BR>
<BR>
<INPUT TYPE=SUBMIT>
</FORM>
</BODY>
</HTML>
```

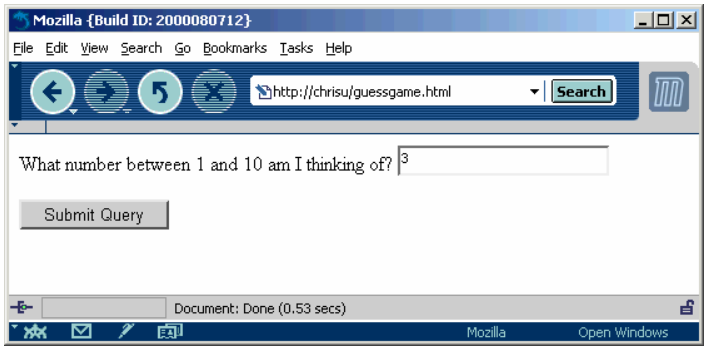
2. Išsaugokite kaip `guessgame.html`.

3. Uždarykite šitą failą ir pradėkite naują:

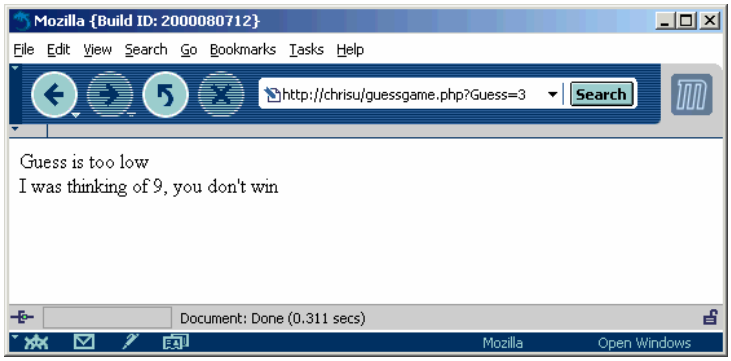
```
<HTML>
<HEAD></HEAD>
<BODY>
<?php
$Number = rand(1,10);
if ($Guess>$Number) {
echo "Guess is too high";
echo "<BR>I was thinking of $Number, you don't ";
}
if ($Guess<$Number) {
echo "Guess is too low";
echo "<BR>I was thinking of $Number, you don't ";
}
?>
win
</BODY>
</HTML>
```

4. Išsaugokite kaip `guessgame.php`.

5. Atidarykite `guessgame.html` savo naršyklėje, bei parašykite skaičių:



6. Paspauskite Submit Query ir žiūrėkite ar pataikėte:



Kaip tai veikia

Mes iš tikro kiek sukčiavome šiame pavyzdyje, nes skaičius generuojamas jau po vartotojo įvesto skaičiaus. Tačiau tai neturi jokios įtakos rezultatui, nes mūsų generuojamas skaičius nepriklauso nuo vartotojo spėtojo. Pirmoji programa paklausia vartotojo skaičiaus, o atsakymą priskiria `Guess` kintamajam.

```
What number between 1 and 10 am I thinking of?
<INPUT NAME="Guess" TYPE="Text">
```

Tada mūsų duomenys persiunčiami `guessgame.php` programai kuri gali apdoroti reikšmę, nes jo reikšmę išsaugota kintamajame `$Guess`, kuris buvo automati6kai sukurtas PHP. Pažiūrėkime paeiliui sekančią programą. Pirmoji linija generuoja skaičių tarp 1 ir 10:

```
$Number = rand(1,10);
```

`Rand` funkcija yra labai paprasta, paprasčiausiai parašote minimalią ir maksimalią reikšmę atskirtą kableliu ir ji sugeneruos atsitiktinį skaičių tarp šių dviejų reikšmių. Rezultatas yra išsaugomas `$Number` kintamajame.

Tada mes paimame skaičių kurį įvedė vartotojas, kuris saugomas `$Guess`, kintamajame ir palyginame kurį „sugalvojo“ PHP. Mes patikriname ar reikšmė, saugoma `$Guess` kintamajame yra didesnė nei reikšmė esanti kintamajame `$Number`. Jei taip ir yra, mes vykdome sekančias kodo eilutes:

```
if ($Guess>$Number) {
echo "Guess is too high";
echo "<BR>I was thinking of $Number, you don't ";
}
```

Kodas tarp riestinių skliaustelių informuoja vartotoja, kad spėjamas skaičius buvo per didelis, ir parašo koks gi iš tikro turėjo būti skaičius. Taip pat jis prideda nepilną "you don't", kuri yra papildoma vėliau.

Antrasis `if` operatorius patikrina ir vykdo odą kai spėtas skaičius buvo per mažas:

```
if ($Guess<$Number) {
echo "Guess is too low";
echo "<BR>I was thinking of $Number, you don't ";
}
```

Šiuo atveju mes informuojame vartotoją, kad spėjimas buvo per mažas, ir vėl gi parašome koks gi turėjo būti skaičius, be to pridedame nepilną frazę "you don't".

Sekanti kodo eilutė užbaigia PHP skriptą, vėliaus mes parašome eilutę su vieninteliu tekstu "win".

```
?>
Win
```

Ši eilutė yra visada parodoma, ir jūs galite pastebėti, kad mes ją naudojame užbaigdami "you don't" frazę. Tačiau, kaip galite spėti, nieko neatsitinka, jei vartotojas atspėja teisingą numerį. Šiuo atveju, nei vienas iš `if` operatorių nėra teisingas – true. Viskas ką mes parodome yra paprastas "win".

Taigi, mes iš tikro netikrinome ar vartotojas atspėjo, mes tikėjome, kad jei skaičius nėra per mažas ir jei nėra per didelis, jis turi būti teisingas. Ką tik jūs susipažinote su operatoriais kurie tikrina lygybę.

== ir === operatoriai

Mes jau naudojome lygybės ženklus atlikdami kiek kitokią operaciją PHP kalboje, jūs tikriausiai jau pastebėjote, kad lygybės ženklas turi dvi skirtingas panaudojimo galimybės. Vienas lygybės ženklas yra **priskyrimo operatorius**, dvigubos lygybės ženklas yra **lygybės operatorius**. Tai labai svarbus skirtumas, jei mes atliksime sekančią operaciją:

```
$Kintamasis = 5;
$Kintamasis = 7;
```

Viršutinė linija priskiria kintamajam `$Kintamasis` reikšmę 5, o tada ši reikšmė yra panaikinama ir kintamajam priskiriama naujoji – 7. Taigi antroji linija perrašo pirmąją.

Visas skirtumas tas, kad lygybės operatorius jokiu būdu neįtakoja kintamojo reikšmės. Sekančioje eilutėje:

```
if ($Kintamasis == 7) echo ("Jūsų kintamojo reikšmė yra
septyni");
```

`$Kintamasis` nėra niekaip pakeičiamas šios operacijos, jis tik palyginamas ar kintamasis lygus ar ne septyniems.

Yra ir kitas lygybės operatorius, įvestas visai neseniai PHP 4.01 versijoje. Jis rašomas su trimis lygybės ženklais ir teisingas bus tik tuo atveju, jei reikšmės ir kintamųjų duomenų tipais sutaps:

```
if ($Kintamasis === $AtsitiktinisSkaicius) echo ("Jūsų
kintamasis sutampa su $AtsitiktinisSkaicius ");
```

The != and <> Operators

Lygybės operatoriaus == priešingybė yra **nelygybės operatorius** !=.

```
 if ($Kintamasis != 7) echo ("Jūsų kintamasis tikrai nėra
septyneri");
```

Yra ir kitas būdas nustatyti nelygybę, pasinaudojant **daugiau nei** ir **mažiau nei** operatoriai. Jie naudojami sekančiu būdu:

```
if ($Kintamasis <> 7) echo ("Jūsų kintamasis tikrai nėra
septyneri");
```

Vienintelis kartas kai šių operatorių rezultatas bus false – neigiamas, kai `$Kintamasis` yra 7.

Pasinaudodami skyriuje prieš tai išdėstyta medžiaga, padarykime pavyzdį. Mes ne tik pateiksime klausimą, bet ir pasakysime ar atsakymas buvo teisingas.

Išbandykite – Lygybės ir nelygybės operatorių naudojimas

1. Naujame faile parašykite sekantį tekstą:

```
<HTML>
<HEAD></HEAD>
<BODY>
<FORM METHOD=GET ACTION="quiz.php">
What is the capital of Portugal?
<BR>
<BR>
<INPUT NAME="Question1" TYPE="Radio" VALUE="Porto">
Porto
<BR>
<INPUT NAME="Question1" TYPE="Radio" VALUE="Lisbon">
Lisbon
<BR>
<INPUT NAME="Question1" TYPE="Radio" VALUE="Madrid">
Madrid
<BR>
<BR>
<INPUT TYPE=SUBMIT>
</FORM>
</BODY>
</HTML>
```

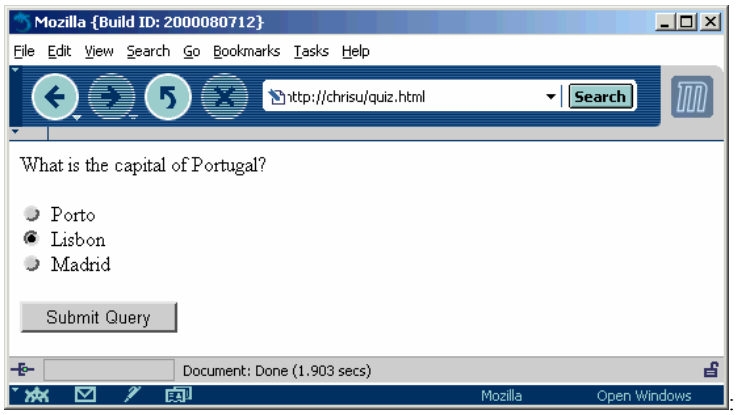
2. Išsaugokite kaip `quiz.html`.

3. Uždarykite senąjį ir atsidarę dar vieną failą parašykite:

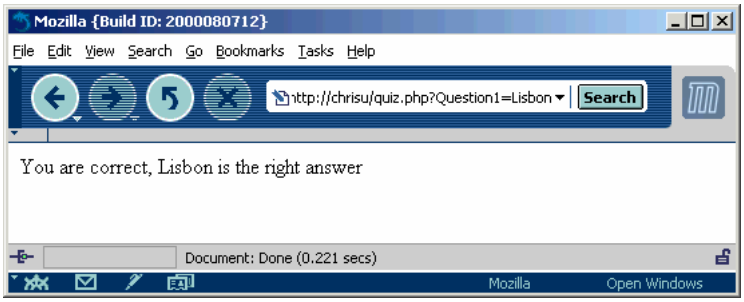
```
<HTML>
<HEAD></HEAD>
<BODY>
<?php
if ($Question1=="Lisbon") echo "You are correct, Lisbon is
the right answer";
if ($Question1!="Lisbon") echo "You are incorrect, Lisbon
is the right answer";
?>
</BODY>
</HTML>
```

4. Išsaugokite kaip `quiz.php`.

5. Atidarykite `quiz.html` ir pateikite atsakymą



6. Jei jūs pasirinkote Lisbon, turėtumėte pamatyti sekantį vaizdą:



7. Grįžkite ir pabandykite pasirinkti kitokį atsakymo variantą, pažiūrėkite koks buvo atsakymas.

Kaip tai veikia

Čia mažai ką tenką aiškinti. Mes apžvelgėme pasirinkimo mygtukus prieš tai buvusiame skyriuje, ir jūs jau turėtumėte žinoti, kad ta opcija kuri yra pasirinkta ir suteikia savo reikšmę `NAME` atributui:

```
<INPUT NAME="Question1" TYPE="Radio" VALUE="Porto">
Porto
<BR>
<INPUT NAME="Question1" TYPE="Radio" VALUE="Lisbon">
Lisbon
<BR>
<INPUT NAME="Question1" TYPE="Radio" VALUE="Madrid">
Madrid
```

Tada mes paimame šią reikšmę, kuri PHP išsaugoma kaip kintamasis `$Question1`, ir mūsų antrojoje programoje palyginame ją su mūsų teisingu atsakymu "Lisbon" norėdami pažiūrėti ar jos sutampa:

```
if ($Question1=="Lisbon") echo "You are correct, Lisbon is
the right answer";
```

Jei sutampa, mes pranešame "You are correct", jei ne, nieko neatsitinka.

PHP toliau nuosekliai vygdo savo programą ir sekančioje eilutėje patikrina ar `$Question1` nesutampa su "Lisbon":

```
if ($Question1!="Lisbon") echo "You are incorrect, Lisbon
is the right answer";
```

Logiškai tariant, vienas iš šių atsakymų turi būti teisingas, arba atsakymas pasirinktas tas kuris ir turi būti arba ne.

Mums beliko tik apžvelgti paskutinių loginių operatorių grupę.

Loginiai operatoriai (AND, OR, NOT)

Loginiai operatoriai nėra tokie baisūs kaip gali pasirodyti iš pirmo žvilgsnio. Tiesa pasakius jų angliškos reiklšmės leidžia suprasti jų naudojimo paskirtį PHP kalboje. Jūs galite pasakyti pavyzdžiui – jei šiandiena šeštadienis ir oras yra geras, aš eisiu į paludimį. Tas pats vyksta ir su PHP:

```
  if ($diena == "Šeštadienis" AND $oras == "Saulėtas") echo
("Laikas į papludimį");
```

`AND` taip pat gali būti rašomas ir su dvigubu 'and' ženklu (`&&`), pavyzdžiui:

```
  if ($diena == "Šeštadienis" && $oras == "Saulėtas") echo
("Laikas į papludimį");
```

`OR` ir `NOT` operatoriai yra lygiai taip pat suprantami koki1 ir turi reikšmę. Mes galime pasinaudoja `OR` operatoriu, pasakyti priešingą loginę reikšmę:

```
  if ($diena == "Pirmadienis" OR $oras == "lyja") echo
("Niekas neis į paplūdimį");
```

Jei yra pirmadienis ir lauke lyja, mes negalime eiti į paplūdimį. `OR` operatorius taip pat gali būti parašytas dvigubu `||` ženklu. Taigi paskutinį kodą buvo galima parašyti ir taip:

```
  if ($diena == "Pirmadienis" || $oras == "lyja") echo
("Niekas neis į paplūdimį");
```

Vienas įdomus pastebėjimas, kuris tačiau neurėtu įtakoti jūsų kodo. && ir || operatoriai turi kiek skirtingą pirmenybę. Jei bus suformuotas loginis klausimas pasinaudojus šiais operatoriais jie turės pirmenybę prieš savo tekstines atitikmenis.

Paskutinis operatorius apie kurį lieka pašnekėti turi tik vieną formą. Jūs negalite tiesiogiai naudoti žodžio `NOT` kaip operatoriaus. `NOT` operatorius vaizduojamas kaip šauktukas, jei jis rašomas skliaustelių išorėje, jis pakeičia priešinga reikšme skliausteliuose esančią reikšmę. Taigi, jei reikšmė teigiama, NOT operatorius ją paverčią neigiama ir atvirkščiai. Pavyzdžiui, jei diena ne šeštadienis, mes negalime eiti į paplūdimį.

```
  if !($diena == "Šeštadienis") echo ("Tada neinam į
paplūdimį šiandien");
```

kaip matote jis turi tokį patį efektą kaip ir neigimo operatorius `!=` kurį mes pristatėme ankščiau. Tiesa pasakius jums nebūtinai reikia turėti kokį nors operatorių `if` eilutėje. Jūs galite patalpinti kintamąjį sąlygos dalyje:

```
  if !($Atsakymas) echo ("Nėra jokio atsakymo");
```

Ši eilutė kaip atsakymą pateiks, kad nėra jokios reikšmės priskirtos `$Answer` kintamajam, arba jei `$Atsakymas` turi nuline reikšmę (kas PHP kalboje tolygu būti tuščiam). Žinote kodėl tai atsitiks? Taip yra todėl, kad `!` operatorius paverčia neigiamąją teigiamą-reikšmę kintamojo `$Atsakymas`, tad jei `$Atsakymas` sugražina neigiamą reikšmę, `!($Atsakymas)` paverčia ją teigiama ir `if` eilutė bus vykdoma.

Yra ir daugiau detalių apie loginius operatorius, bet padarykime pertrauką pavyzdžiui. Mūsų programa bus skirta mašinų nuomos kompanijai, kurios pagalba ji galės apskaičiuoti ar klientas gali vairuoti vieną iš jų mašinų. Norint tai padaryti reikia galiojančių teisių ir turėti daugiau nei 21 metus. Mūsų programa patikrins šias ir keletą kitų detalių.

Išbandykite – Loginių operatorių naudojimas

1. Kaip jau turbūt žinote – atsidarykite teksto redagavimo programą ir parašykite:

```
<HTML>
<HEAD></HEAD>
<BODY>
<B>Namllu Car Hire Company</B>
<FORM METHOD=POST ACTION="car.php">
First Name:
<INPUT NAME="FirstName" TYPE="Text">
Last Name:
<INPUT NAME="LastName" TYPE="Text">
Age:
<INPUT NAME="Age" TYPE="Text"SIZE="3">
<BR>
<BR>
Address:
<TEXTAREA NAME="Address" ROWS=4 COLS=40>
</TEXTAREA>
<BR>
<BR>
Do you hold a current driving license?
<INPUT NAME="License" TYPE="Checkbox">
<BR>
<BR>
<INPUT TYPE=SUBMIT VALUE="Click here to Submit
application">
</FORM>
</BODY>
</HTML>
```

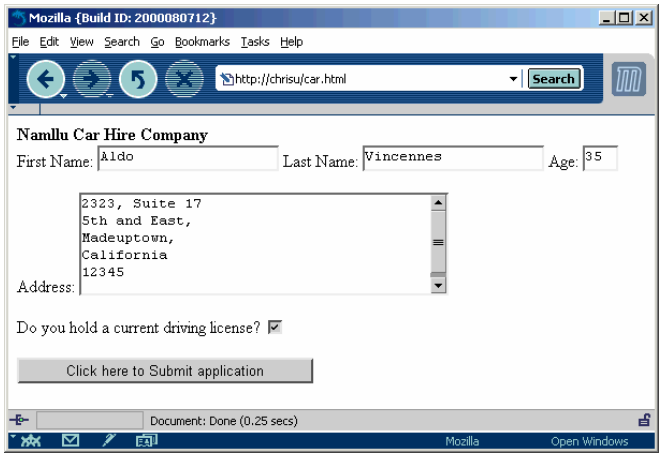
2. Išsaugokite kaip `car.html`.

3. Atsidarykite naują ir parašykite:

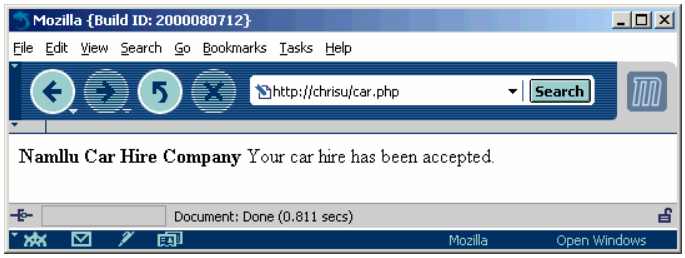
```
<HTML>
<HEAD></HEAD>
<BODY>
<B>Namllu Car Hire Company</B>
<?php
if ($Age>20 AND $License=="on") echo ("Your car hire has
been accepted.");
if ($Age<21 OR $License=="") echo ("Unfortunately we cannot
hire a car to you.");
?>
</BODY>
</HTML>
```

4. Išsaugokite kaip `car.php`.

5. Atidarykite `car.html` savo naršyklėje ir įveskite duomenis:



6. Paspauskite ant Submit application norėdami pamatyti rezultatą:



Kaip tai veikia

HTML elementų kaip matote yra daug tačiau mes naudojame tiktai du iš jų, tikriname amžių ir vairuotojo teisių turėjimą.

```
<INPUT NAME="Age" TYPE="Text"SIZE="3">
<BR>
<BR>
Address:
<TEXTAREA NAME="Address" ROWS=4 COLS=40>
</TEXTAREA>
<BR>
<BR>
Do you hold a current driving license?
<INPUT NAME="License" TYPE="Checkbox">
```

Teksto laukelio `NAME` atributas pavadintas "Age", taigi `$Age` kintamasis yra sukuriamas ir jame saugojamas vartotojo įvestas amžius. Sekantis elementas klausiantis dėl vairuotojo teisių turėjimo gali būti pažymėtas arba ne. Taigi `$License` kintamasis gali turėti "on" reikšmę arba neturėti jokios.

Reikšmė "on" priklauso nuo naršyklės kurią jūs naudojate, bet atsižvelgiant į tai, kad dauguma vartotojų naudojasi Internet Explorer, Netscape Navigator ir Opera naršyklėmis neturėtu kilti problemų.

Mes panaudojame abu šiuos kintamuosius mūsų PHP kode antrajame faile. Pirmoji linija `car.php` faile nurodo, kad jei vartotojas turi daugiau nei 20 metų, bei vairuotojo teises, mes galime išnuomoti jam mašiną:

```
  if ($Age>20 AND $License=="on") echo ("Your car hire has
been accepted.");
```

Antroji linija sako priešingai:

```
  if ($Age<21 OR $License=="") echo ("Unfortunately we cannot
hire a car to you.");
```

Arba vartotojo amžius mažiau nei 21, arba neturi teisių ir mes atsisakome išnuomoti jam mašiną. Tai tiek šios programos.

Vienos nenumatytos galimybės mes neaptarėme – kas atsitiks jei vartotojas įves metų skaičių tarp 20 ir 21, pavyzdžiui 20.5? Taip mes galime gauti net du atsakymus – kad nuoma yra galima ir, kad ne. Norint ištaisyti šią ‚skylę' mes turime kiek pataisyti skriptą ir tam pristatysime dar vienus loginius operatorius.

>= ir <= operatoriai

Šie operatoriai jums turėtu atrodyti jau gerai pažystami, nes jie abu tėra ankščiau nagrinėtu operatorių kombinacijos. Jei norite pasakyti, kad skaičius turi būti **mažesnis nei arba lygus** kažkokiai reikšmei, tada naudokite <= operatorių. Tas pats pritaikoma ir **daugiau nei arba lygu** operatoriui >=. Nors jie turi tik vieną lygybės ženklą, jie neatlieka priskyrimo operacijos ir yra naudojami išskirtinai palyginimui. Norint, kad mūsų paskutinis pavyzdys veiktu taip kaip ir turi veikti, mes turime pakeisti pirmąją liniją `car.php` failo:

```
  if ($Age>=21 AND $License=="on") echo ("Your car hire has
been accepted.");
```

Operatorių kombinavimas

Mes jau kombinavime loginius operatorius su lygybės operatoriais, tačiau nėra PHP kalboje nėra jokių apribojimų kiek operatorių gali būti vienoje if išraiškoje. Perfrazuojant mūsų pirmąjį klausimą – "jei oras yra geras ir diena yra šeštadienis, mes eisime į paplūdimį", galima pasakyti ir kitaip "jei diena ne pirmadienis, antradienis, trečiadienis, ketvirtadienis, penktadienis ar sekmadienis ir nelyja mes eisime į paplūdimį":

```
  if ($diena != "pirmadienis")
  OR ($diena !="antradienis")
  OR ($diena !="trečiadienis")
  OR ($diena !="ketvirtadienis")
  OR ($diena !="penktadienis")
  OR ($diena !="šeštadienis")
  OR ($oras!="Lyja")) echo ("Į paplūdimį!");
```

Vėl gi tai panašu į mūsų kalbą tik labai pedantiška. Nėra limito kaip mes galime kombinuoti šiuos operatorius, ar kiek daug jūs galite parašyti, paprasčiausiai gali būti kiek sudėtinga suprasti ką loginė išraiška reiškia jei ją dažnai naudojate. Kaip manote ką sako sekantis kodas?

```
  if (($diena == "Pirmadienis" AND $menesis != "Balandis")
  OR ($diena == "Antradienis" AND $laikas != "12")
  OR !($menesis != "Gruodis")) echo ("Paskirtas tarybos
susitikimas")
```

Jis sako: jei ne Pirmadienis ir ne Balandis, arba jei Antradienis ir jei ne 12.00, arba ne Gruodis, tada mes turime paskirta tarybos susitikimą. Jūs žinoma tai suprasite, tačiau tai užims laiko ir pastangų.

Pažiūrėkime į kitą pavyzdį, kuris apskaičiuotu mašinos draudimo įmoką, remdamasi duotais keturiais kintamaisiais – vairuotojo amžiumi, mašinos verte, mašinos maksimaliu greičiu ir jos variklio tūriu. Šioje neegzistuojančioje draudimo bendrovėje yra galimi tik trys draudimo variantai. Pirmasis variantas yra Visapusis draudimas kainuojantis $1500, kuris prieinamas šioms rizikos grupėms:

- vairuotojui mažiau nei 25 metai
- mašinos vertė didesnė nei $10,000

- ❑ mašinos variklis didesnis nei 1.5L
- ❑ mašinos maksimalus greitis daigiau nei 200 kilometrų per valandą

Standartinis draudimo variantas - $1000, taikomas tiems kurie nepatenka į nei vieną viršuje išvardytą rizikos grupę.

Trečioji yra skirta pensijinio amžiaus žmonėms - $750, kuri taikoma šiais atvejais:

- ❑ vairuotojui daugiau nei 65,
- ❑ mašinos vertė neviršija $5,000, arba maksimalus greitis ne didesnis nei 140 km/h
- ❑ žmogus nepatenka į nei vieną aukščiau išvardytą rizikos grupę

Mūsų puslapis apskaičiuos pagal pateiktus kriterijus koks draudimo variantas turi būti pasiūlytas.

Išbandykite – Operatorių kombinacijų naudojimas

1. Dar kartą atidarykite savo teksto redaktorių ir parašykite:

```
<HTML>
<HEAD></HEAD>
<BODY>
<B>PHP Draudimo bendrovė</B>
<BR>
<BR>
<FORM METHOD=POST ACTION="quote.php">
Koks jūsų amžius?
<INPUT TYPE=TEXT NAME="amzius" SIZE=3>
<BR>
<BR>
Koks jūsų mašinos maksimalus greitis?
<INPUT TYPE=TEXT NAME="greitis">
<BR>
<BR>
Kokia apytiksli jūsų mašinos vertė?
<SELECT NAME="kaina">
<OPTION VALUE=5000>Mažiau nei $5,000</OPTION>
<OPTION VALUE=7000>Tarp $5,000 ir $7,000</OPTION>
<OPTION VALUE=10000>Tarp $7,000 ir $10,000</OPTION>
<OPTION VALUE=25000>Daugiau nei $10,000</OPTION>
</SELECT>
<BR>
<BR>
Koks jūsų mašinos variklio tūris?
<SELECT NAME="variklioturis">
<OPTION VALUE=1.0>1.OL</OPTION>
<OPTION VALUE=1.3>1.3L</OPTION>
<OPTION VALUE=1.5>1.5L</OPTION>
<OPTION VALUE=2.0>2.0L</OPTION>
</SELECT>
<BR>
<BR>
<INPUT TYPE=SUBMIT VALUE="Paspauskite čia">
</FORM>
</BODY>
</HTML>
```

2. Išsaugokite kaip `quote.html`.

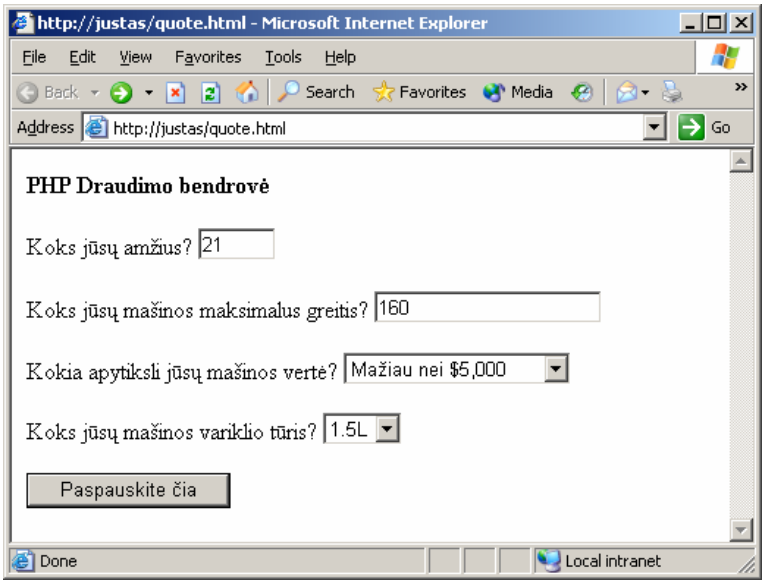
3. Uždarykite pirmą failą ir atsidarę naują, parašykite:

```
<HTML>
<HEAD></HEAD>
<BODY>
```

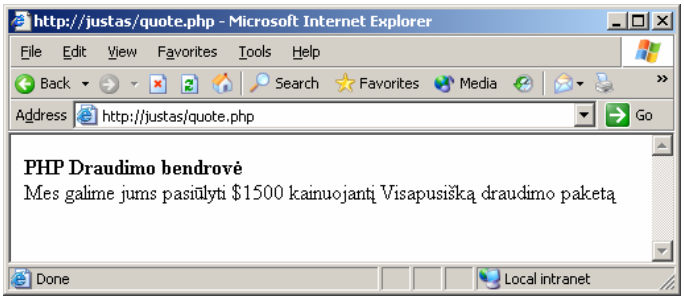
```
  <B> PHP Draudimo bendrovė </B><br>
  <?php
  if ($amzius<25 OR $greitis>200 OR $kaina>10000 OR
$variklioturis>1.5)
  {
  echo ("Mes galime jums pasiūlyti $1500 kainuojantį
Visapusišką draudimo paketą");
  }
  if ($amzius>=65 AND ($kaina<=5000 OR $greitis<=140) AND
$kaina<=10000 AND
  $variklioturis<=1.5 AND $greitis<=200)
  {
  echo ("Mes galime jums pasiūlyti $750 kainuojantį
Pensininkų graudimo paketą");
  }
  if (($amzius<65 OR $kaina>5000 AND $greitis>140) AND
$amzius>=25 AND $greitis<=200 AND $kaina<=10000 AND
$variklioturis<1.5)
  {
  echo ("Mes galime jums pasiūlyti $1000 kainuojantį
Standartinį draudimo paketą");
  }
  ?>
  </BODY>
  </HTML>
```

4. Išsaugokite kaip `quote.php`.

5. Atidarykite `quote.html`



6. Padarykite tai ką sako mygtukas:



Kaip tai veikia

Taip yra ir lengvesnių būdu atlikti šiai užduočiai, nei mūsų pasirinktas, tačiau mes norėjome parodyti kaip sukombinuoti daug kombinacijų vienoje išraiškoje. Pirmoji programa `quote.html` yra paprasta HTML forma kuri paklausia jūsų duomenų ir juos išsaugo. Šiame pavyzdyje mes pasinaudojome visais vartotojo įvestais duomenimis. Mes nenagrinėsime `quote.html` detaliau, tik pažymėsime, kad buvo sukurti tokie kintamieji - `$amzius`, `$greitis`, `$variklioturis`, ir `$kaina`.

Mus labiau domina kompleksiniai sąlygos variantai, kurie turėjo veikti `quote.php` faile. Iš viso mes turim atlikti tris testus, ir įsitikinti, kad visi kandidatai kurie naudosis šia programa gaus atsakymą ir **tik vieną** atsakymą.

Pirmasis testas skirtas visapusiško draudimo paketui. Yra keturios sąlygos, ir jei bent viena jų yra „teisinga“ vartotojui gali būti rekomenduota tik pats brangiausias draudimo paketas. Mes patikriname kintakąjį `$amzius`, norėdami sužinoti ar jis mažesnis nei 25, kintamąjį `$greitis`, norėdami pažiūrėti ar jis didesnis nei 200, kintamąjį `$variklioturis`, norėdami sužinoti ar variklio tūris didesnis nei 1.5L, ir mašinos vertę – ar ji didesnė nei $10,000. Jei nors viena iš sąlygų patenkinta mes pateikiame atsakymą apie siūlomą visapusiško draudimo paketą.

```
  if ($amzius<25 OR $greitis>200 OR $kaina>10000 OR
$variklioturis>1.5)
  {
  echo ("Mes galime jums pasiūlyti $1500 kainuojantį
Visapusišką draudimo paketą");
  }
```

Antra, mes patikriname dėl atitikimo pensijinio draudimo paketui. Jis taikomas kai `$amzius` didesnis nei 65,o `$kaina` yra 5000 ar mažiau, arba `$greitis` yra 140 arba mažesnis. Mes atliekame antrąjį testą atskirtą skliausteliais, nes tik viena iš reikšmių turi būti teisinga, nors gali būti ir abi. Mes taip pat turime patikrinti, kad vartotojas būdamas pensininku, nepakliūtu į kitas rizikos grupes, nes jas atitinkantiems klientams jau buvo pasiūlyti brangesni draudimosi paketai.

```
  if ($amzius>=65 AND ($kaina<=5000 OR $greitis<=140) AND
$kaina<=10000 AND
  $variklioturis<=1.5 AND $greitis<=200)
  {
  echo ("Mes galime jums pasiūlyti $750 kainuojantį
Pensininkų graudimo paketą");
  }
```

Galiausiai, norėdami „pagauti“ visus kitus variantus mes patikriname ar `$amzius` yra tarp 25 ir 64, mašinos vertė didesnė nei 5000, ir maksimalus greitis didesnis nei 80. tai reiškia, kad mūsų vairuotojas nepatenka į pensijinio amžiaus grupę, bet taip pat mes turime patikrinti, kad jam/jai viršijus kitus kriterijus būtu pasiūlytas brangiausias draudimo paketas. Taigi mes tikriname `$greitis`, `$kaina`, and `$variklioturis` kintamuosius.

```
  if (($amzius<65 OR $kaina>5000 AND $greitis>140) AND
$amzius>=25 AND $greitis<=200 AND $kaina<=10000 AND
$variklioturis<1.5)
  {
  echo ("Mes galime jums pasiūlyti $1000 kainuojantį
Standartinį draudimo paketą");
  }
```

Daugybinės sąlygos – else ir elseif

Mes apžvelgėme daugybines sąlygos variantus su vienu `if` elementu, bet kas atsitiktu jei mes norėtumėme vykdyti komandą kai sąlyga yra teigiama ir kitą komandą kai neigiamą? Jūs visada galite pakeisti sąlygą ir rašyti `if` elementą, bet yra daug lengvesnis būdas tai padaryti. `else` elementas dirba taip:

```
if (sąlyga yra patenkinta)
{
atliekamos komados šiuose sklaisuteliuose
}
else
{
atliekamos komados šiuose sklaisuteliuose
}
```

Mes galėtumėme perdaryti mūsų mašinos nuomos pavyzdį dar kartą:

```
if ($Age>=21 AND $License=="on")
  {
    echo ("Your car hire has been accepted.");
  }
    else
  {
    echo ("Unfortunately we cannot hire a car to you.");
  }
```

Atkreipkite, kad mes komandų eilutę pradedame iš naujos eilutės, tai nėra privaloma norint, kad mūsų pavyzdys veiktu, bet taip yra lengviau skaityti.

Ženkime dar vieną žingsnį, kas atsitiks jie jūs norėsite patikrinti kintamąjį su keletu kitu, o ne vienu? Tai padės atlikti `elseif` elementas, leidžiantis sąlygą tikrinti keletą kartų:

```
if (kintamasis yra lygus reiksme1)
  {
    atliekamos komandos šiuose skliausteliuose
  }
elseif (kintamasis yra lygus reiksme2)
  {
    atliekamos komandos šiuose skliausteliuose
  }
else
  {
    atliekamos komandos šiuose skliausteliuose
  }
```

Mes galime dar labiau patobulinti mūsų mašinos nuomos programą, įvesdami atsakymą, kad visi tarp 18 ir 21 gali išsinuomoti mašiną, jei pateiks garantuojančio asmens patvirtinimą:

```
if ($Age>=21 AND $License=="on")
  {
    echo ("Your car hire has been accepted.");
  }
    elseif ($Age>=18 AND $License=="on")
  {
    echo ("Your car hire has been accepted, subject to you
providing the name  of a guarantor.");
  }
else
  {
    echo ("Unfortunately we cannot hire a car to you.");
  }
```

Tai suderina tris galimas sąlygas, trečioji yra `else`, kuri „pagauna“ visus variantus jei pirmi trys nebuvo patenkinti. Niekas nemaišo mums pridėti dar ir dar `elseif` elementų į mūsų programą. Pavyzdžiui programa apskaičiuojanti egzaminų rezultatus, kuriuose galima gauti A, B, C, D ar E, gali būti pavaizduota štai taip:

```
if ($grade>70) echo "You got an A";
elseif ($grade > 60) echo "You got a B";
elseif ($grade > 50) echo "You got a C";
elseif ($grade > 40) echo "You got a D";
elseif ($grade > 30) echo "You got an E";
else echo "You failed";
```

Vis tiek tai neatsako į vieną klausimą; kaip padaryti kad vieno palyginimo rezultatas butų panaudotas kitame palyginime. Pavyzdžiui, jūs norite patobulinti egzaminų tikrinimą, patikrindami ar asmuo gavęs A įvertinimą taip pat nusipelno specialaus įvertinimo už savo projektą. Tokiu atveju jums reikia patalpinti vieną elementą `if` kitame.

Įterpti If elementą vienas į kitą

Vieno `if` elemento įterpimas į kitą vadinamas **įterpimu** (nesting). Taigi jei mes patobulintumėme mūsų kodą, mes galėtumėme įterpti `if` elementą į mūsų 'A grade' `if` elementą.

```
if ($Grade>70)
  {
    echo ("You got an A.");
    if ($ProjectGrade>70)
      {
        echo ("You also got a special merit");
      }
  }
```

Jūs galite tai padaryti ir su keletu `AND` operatorių, bet tokį kodą sunkiau skaityti. Jus nereikia sustoti šioje vietoje, nes įterpimo procedūra gali būti praktiškai begalinė, bet vėl gi jūsų kodą būtu sunku skaityti ir suprasti. Mes galime dar patobulinti šį kodą, patikrindami ar lankomumas buvo šimtaprocentinis ir jei tai apdovanoti mokinį dėl lankomumo:

```
if ($Grade>70)
  {
    echo ("You got an A.");
    if ($ProjectGrade>70)
      {
        if ($AttendanceRecord==100)
          {
            echo ("You also got a special distinction");
          }
        else
          {
            echo ("You also got a merit");
          }
      }
  }
```

Kaip matote skliaustelių kuriuos reikia atidaryti ir uždaryti skaičius vis didėja, ir jei jūs parašysite jų klaidingą skaičių – pavyzdžiui turėsite trys atidarytus ir tik du uždarytus skliaustelius, kodas neveiks ir bus pranešta apie klaidą. Jums labai atidžiai reikia sekti šiuos skaičius. Tai daro labai svarbiu kodo išdėstymą. Jo rašymo stilius niekaip neįtakos programos veikimo, tačiau jums tikrai bus lengviau dirbant su didelėm kodo dalim. Pažvelkite į sekantį kodą, kur viskas surašyta be išdėstymo, ar jis tikrai aiškus iš pirmo žvilgsnio?

```
if ($Grade>70)
{
```

```
echo ("You got an A.");
if ($ProjectGrade>70)
{
if ($AttendanceRecord==100)
{
echo ("You also got a special distinction");
}
else
{
echo ("You also got a merit");
}
}
```

Tiesa pasakius aš specialiai praleidau vienus skliaustelius! Oops! Nedarykite taip ☺

Padarykime pavyzdį kuriame būtu naudojamos visos ankščiau išvardytos galimybės. Mes darysime atostogų rezervavimo formą keletui maršrutų. Jis paskaičiuos kainas priklausomai nuo viešbučio lygio ir jūsų tikslo. Bus trys galimos vietos – Praha, Barselona ir Viena, į kiekviena iš kurių palaipsniui didėja kainos. Taip pat kiekviename mieste bus dviejų rūšių viešbučiai – trijų ir keturių žvaigždučių. Tačiau kelionių kompanijos yra kelionių kompanijos ir pas juos kaina nebūtinai kyla tolygiai. Tad skirtumas tarp trejų ir keturių žvaigždučių viešbučių Barselonoje $1500 , tuo tarpu kai skirtumas tarp tokio pačio tipo viešbučių Vienoje bus net $2250! Mūsų PHP programa turės atsižvelgti į visa tai.

Išbandykite – sudėtingų kompleksinių sąlygų naudojimas

1. Atidarykite savo teksto redaktorių ir rašykite:

```
<HTML>
<HEAD></HEAD>
<BODY>
<B>Namllu Holiday Booking Form</B>
<FORM METHOD=GET ACTION="holiday.php">
Where do you want to go on holiday?
<BR>
<BR>
<INPUT NAME="Destination" TYPE="Radio" VALUE="Prague">
Prague
<BR>
<INPUT NAME="Destination" TYPE="Radio" VALUE="Barcelona">
Barcelona
<BR>
<INPUT NAME="Destination" TYPE="Radio" VALUE="Vienna">
Vienna
<BR>
<BR>
What grade of hotel do you want to stay at?
<BR>
<BR>
<INPUT NAME="Grade" TYPE="Radio" VALUE="Three">
Three Star
<BR>
<INPUT NAME="Grade" TYPE="Radio" VALUE="Four">
Four Star
<BR>
<BR>
<INPUT TYPE=SUBMIT>
</FORM>
</BODY>
</HTML>
```

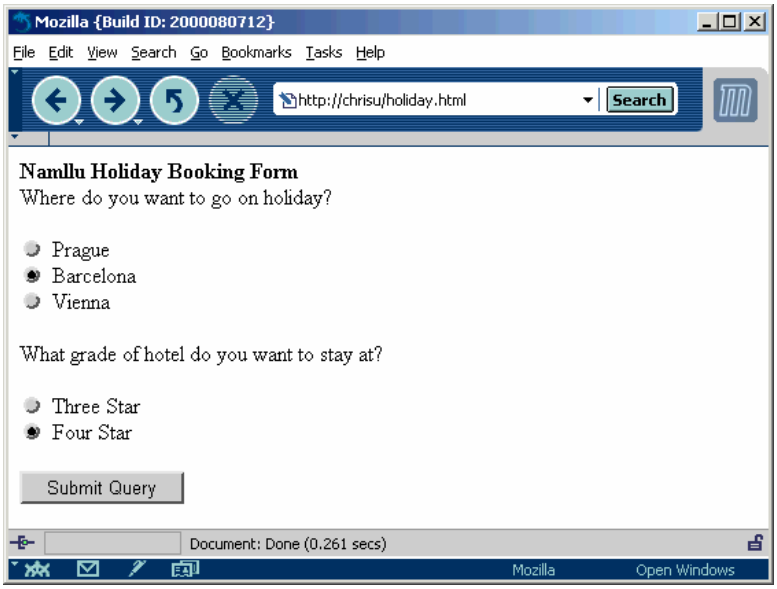
2. Išsaugokite kaip `holiday.html`.

3. Uždarykite failą ir pradėkite naują:

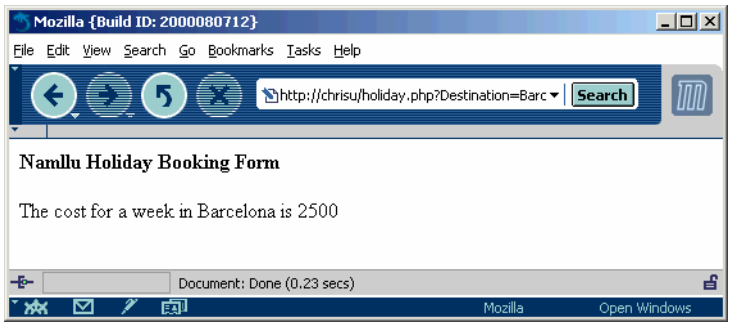
```
<HTML>
<HEAD></HEAD>
<BODY>
<B>Namllu Holiday Booking Form</B>
<BR>
<BR>
<?php
$Price=500;
$StarModifier=1;
$CityModifier=1;
if ($Grade=="Three")
  {
   if ($Destination=="Barcelona")
  {
    $CityModifier=2;
    $Price = $Price * $CityModifier;
    echo "The cost for a week in $Destination is $Price";
  }
elseif ($Destination=="Vienna")
  {
    $CityModifier=3.5;
    $Price = $Price * $CityModifier;
    echo "The cost for a week in $Destination is $Price";
  }
elseif ($Destination=="Prague")
  {
    $Price = $Price * $CityModifier;
    echo "The cost for a week in $Destination is $Price";
  }
else
  {
    echo ("You've not entered a value for destination, go
back and do it again");
  }
}
elseif ($Grade=="Four")
  {
    $StarModifier=2;
    if ($Destination=="Barcelona")
      {
        $CityModifier=2.5;
        $Price = $Price * $CityModifier * $StarModifier;
        echo "The cost for a week in $Destination is
$Price";
       }
elseif ($Destination=="Vienna")
   {
    $CityModifier=4;
    $Price = $Price * $CityModifier * $StarModifier;
    echo "The cost for a week in $Destination is $Price";
   }
elseif ($Destination=="Prague")
   {
    $Price = $Price * $CityModifier * $StarModifier;
    echo "The cost for a week in $Destination is $Price";
   }
else
  {
    echo ("You've not entered a value for destination, go
back and do it again");
  }
}
else
  {
    echo ("You've not entered a value for hotel grade, go
back and do it again");
}
?>
</BODY>
</HTML>
```

4. Išsaugokite kaip `holiday.php` prieš tai atidžiai jį patikrinę. Įsitikinkite, kad viskas parašyta taip kaip reikia.

5. Atidarykite `holiday.html` savo naršyklėje:



6. Paspauskite ant Submit Query norėdami pamatyti kainą:



Kaip tai veikia

Sukuriami du kintamieji, `$Destination` and `$Grade`, kurie turi vartotjo pasirinkimą apie kelionės tikslą ir viešbučio tipą. Antroji programa, `holiday.php`, yra ilgiausia PHP ptograma kurią mes iki šiol parašėme. Tačiau joje nėra nieko apie mes nebūtumėme šnekėja iki šiol, tad mes pažingsniui ją peržvelgsime ir jūs suprasite, kad joje nieko sudėtingo nėra.

Pirmosios tris linijos sukuria tris reikšmes su nustatytomis vertėmis:

```
$Price=500;
$StarModifier=1;
$CityModifier=1;
```

Dviejų kintamųjų–modifikatorių egzistavimas yra skirtas keisti brangesnių viešbučių ir kelionės tikslų kainą. Jei mes naudosimės keturių žvaigždučių viešbučiu mums teks kaina padidinti dvigubai. Jei apsistosime Vienoje kaina teks padidinti tris kartus ir t.t. Norėdami tai padaryti mes padauginsime mūsų visą kainą iš šių dviejų dviejų kintamųjų reikšmės. Savaitė Prahoje trijų žvaigždučių

viešbutyje kainuoja 500 dolerių ir nereikalauja jokių pakeitimų, tad mes paliekame jiems reikšmę 1; viskas kitas bus brangiau. Taigi mūsų pirmoji operacija patikrina viešbučio lygį:

```
if ($Grade=="Three")
```

Jei jis lygus trims, mes pereiname prie sekančio testo norėdami nustatyti tikslą:

```
if ($Destination=="Barcelona")
```

mes patikriname ar vartotojo pasirinkimas sutampa su Barcelona, ir jei ne, mes pesirenkame kitą skaičiavimų kelią:

```
$CityModifier=2;
$Price = $Price * $CityModifier;
echo "The cost for a week in $Destination is $Price";
```

Visų pirma mes pakeitėme kintamojo `$CityModifier` reikšmę. Tada padauginome jo kainą sekančioje eilutėje, ir gavome naująją kainą. Tada parodome kiek kainuoja išvyka į Barcelona. Atkreipkite dėmesį, kad šitos kodo eilutės bus atliekamos tik tuo atvėju jei pasirinktas trijų žvaigždučių viešbutis, o kelionės tikslas - Barcelona. Tai užbaigia mūsų pirmojąjį įterptą `if` elementą. Antrasis testas patikrina ar tikslas yra Vienna:

```
elseif ($Destination=="Vienna")
```

Visa tai yra viena didelio įterpto `if` elemento dalis. Mes atliekame sekanti koda tik tuo atvėju jei lygis yra trys žvaigždutės ir kelionės tikslas yra Vienna:

```
$CityModifier=4;
$Price = $Price * $CityModifier;
echo "The cost for a week in $Destination is $Price";
```

$`CityModifier` kintamasis padidinamas iki 4, taip atsižvelgiant į padidėjusią kainą priklausomai nuo miesto. Mes padauginame jį iš `$Price` kintamojo sekančioje eilutėje ir pateikiame rezultatus trečiojoje.

Tai pabaigia mūsų antrają veiksmų kriptį.

Mūsų trečiasi tikrinimas skirtas nustatyti ar pasirinkta Praha ar ne:

```
elseif ($Destination=="Prague")
```

Jei taip yra, mes padauginame kaina iš egzistuojančio `$CityModifier` kintamojo, kurio reikšmė šiuo metu yra 1. Mes taip pat pateikiame kelionės tikslą ir kainą, bet vėl gi tik tuo atvėju jei kelionės tikslas yra Praha ir pasirinktas trijų žvaigždučių viešbutis:

```
$Price = $Price * $CityModifier;
echo "The cost for a week in $Destination is $Price";
```

Tada mes įtraukeme eilutę pakreipiančius kodo veiksmus tuo atvėju jei vartotojas nepasirinko kelionės tikslo. Jei jie nepasirinko Prahos, Barselonos ar Vienos, mes turime suprasti, kad jie nepasirinko nieko ir pranešame apie tai:

```
else
  {
    echo ("You've not entered a value for destination, go
back and do it again");
  }
```

Tai užbaigia pusę mūsų programos, tačiau mes turime atlikti tas pačias operacijas jei vartotojas pasirinko keturių žvaigždučių viešbutį. Šiuo atvėju mes naudosime ir viešbučių lygio kintamąjį, padaugine jį iš dviejų:

```
elseif ($Grade=="Four")
  {
    $StarModifier=2;
```

Tada mes patikriname paeiliui kiekvieną miestą, jei reikia padidindami kelionės kainą ir visą tai dar padaugindami iš viešbučių lygio kintamojo. Pavyzdžio kodas aptarnaujantis Barseloną atrodo taip:

```
if ($Destination=="Barcelona")
 {
  $CityModifier=2.5;
  $Price = $Price * $CityModifier * $StarModifier;
  echo "The cost for a week in $Destination is $Price";
 }
```

Kodas kiekvienam tiksliui yra gana panačus, tad mes nenagrinėsime jo toliau. Mes tik vėl pažiurėsime „gaudančią“ kodo eilutę, kuri pasirodo kai nepasirinktas nei vienas miestas. Jums gali iškilti klausimas, kam tai daryti jei vienas toks kodas jau buvo parašytas. Tačiau ankščiau buvęs kodas veikė tik tad, kai pasirinktas trijų žvaigždučių viešbutis, o dabar mes dirbame su keturių:

```
else
  {
   echo ("You've not entered a value for destination, go
back and do it again");
  }
```

Pagaliau mes turime patikrinti, gal kas nors nepasirinko viešbučio lygio:

```
else
 {
   echo ("You've not entered a value for hotel grade, go
back and do it again");
}
```

Tai buvo didelis, daugybės skliaustu reikalaujantis kodas. Tačiau yra alternatyvi struktūra PHP kalboje, kuri leidžia pašalinti skliaustelius kei naudojama daug sąlygų, o ir pats kodas atrodo daug suprantamesnis.

Switch elementas

`Switch` elementas atlieka panašią funkciją kaip ir `elseif` struktūroje, kurią mes nagrinėjome prieš tai. Tačiau tai daro daug glausčiau ir paprasčiau skaityti, bei leidžia atsisakyti beveik visų erzinančių skliaustelių.

Jei mes pažvelgsime į mūsų pažymių pavyzdį jis gali atlikti tas pačias funkcijas tik panaudojant `switch` elementą:

```
switch ($Grade) {
      case $Grade>70:
         echo ("You got an A.");
         break;
      case $Grade>60:
         echo ("You got a B.");
         break;
      case $Grade>50:
         echo ("You got a C.");
         break;
      case $Grade>40:
         echo ("You got a D.");
         break;
      case $Grade>25:
         echo ("You got an E. ");
         break;
     default:
         echo ("You failed");
}
```

Tai nesutaupė labai daug kodo linijų, bet pašalino kodo dalį. Vietoj `if` ir `elseif` dabar mes turime tik vieną `case` elementą ir už jo sekančius veiksmus. Kiekvienu atvėju PHP programa vygdo kiek kitokį kodą.

Jūs pastebėjote, kad mes panaudojome komandą `break` mūsų `switch` elemente. Kai PHP pasiekia `break`, ji nutraukia tai ką daro, palieka visą `switch` struktūrą, ir pereina prie toliau sekančios programinės eilutės. Programa nebetikrina atitikimo su kitais kriterijais, net jei 80 procentų pažimys atitiktu visus kriterijus. Tai yra naudinga, nes mes neturine rašyti begaliuniu, mažų `case` „gaudytojų" kiekvienai situacijai. Jei jūs norite, kad būtu patikrinti visi kriterijai, paprasčiausiai praleiskite konamdą `break`, mors reikia pažymėti, kad ši komanda veikia tik su `switch` elementu, ir neveikia su `if`. Jei jūs praleisite žodį breake, tada visi atvėjai bus teisingi ir jūs gausite A, B, C, D, ir E:

```
switch ($Grade) {
    case $Grade>70:
       echo ("You got an A.");
    case $Grade>60:
       echo ("You got a B.");
    case $Grade>50:
       echo ("You got a C.");
    case $Grade>40:
       echo ("You got a D.");
    case $Grade>25:
       echo ("You got an E. ");
   default:
     echo ("You failed");
}
```

`switch` elementas pristato ir įdomų mažą sutrumpinimą:

```
switch ($State) {
    case "IL":
       echo ("Illinois");
       break;
    case "GA":
       echo ("Georgia");
       break;
    default:
       echo ("California");
       break;
}
```

Jei jūs nurodėte reikšmę prie `case` elemento, ji yra automatiškai tkrinama su kintamuoju kuris yra pateiktas `switch` elementu, norin sužinoti ar jie sutampa, nesvarbu ar jis būtu skaitinis ar raidinis. Atkreipkite dėmesį, kad po `case` mes naudojame dvitaški, o ne kabliataški.

Kadandi jūs jau pakankamai geri suprantate ką daro swich elementas, grįžkime prie mūsų ankstesnio pavyzdžio ir panaudokime `switch`.

Išbandykite – switch elemento naudojomas

1. Grįžkite prie `holiday.php`, ištrinkite jo turinį ir parašykite sekanti tekstą:

```
<HTML>
<HEAD></HEAD>
<BODY>
<B>Namllu Holiday Booking Form</B>
<BR>
<BR>
<?php
$Price=500;
$StarModifier=1;
$CityModifier=1;
$DestGrade = $Destination.$Grade;
switch($DestGrade) {
   case "BarcelonaThree":
      $CityModifier=2;
      $Price = $Price * $CityModifier;
```

```
        echo "The cost for a week in $Destination is $Price";
        break;
    case "BarcelonaFour":
        $CityModifier=2;
        $StarModifier=2;
        $Price = $Price * $CityModifier * $StarModifier;
        echo "The cost for a week in $Destination is $Price";
        break;
    case "ViennaThree":
        $CityModifier=3.5;
        $Price = $Price * $CityModifier;
        echo "The cost for a week in $Destination is $Price";
        break;
    case "ViennaFour":
        $CityModifier=3.5;
        $StarModifier=2;
        $Price = $Price * $CityModifier * $StarModifier;
        echo "The cost for a week in $Destination is $Price";
        break;
    case "PragueThree":
        $Price = $Price * $CityModifier;
        echo "The cost for a week in $Destination is $Price";
        break;
    case "PragueFour":
        $StarModifier=2;
        $Price = $Price * $CityModifier * $StarModifier;
        echo "The cost for a week in $Destination is $Price";
        break;
    default:
        echo ("Go back and do it again");
        break;
}
?>
</BODY>
</HTML>
```

2. Išsaugokite kaip `holiday.php`.

3. Atidarykite `holiday.html` pažymėkite savo pasirinkimą ir jis veiks kai ankčiaus:

Kaip tai veikia

Mes iš karto pateikėme pagyrų `switch` elementui – first, jį lengviau skaityti, ir antra, jis naudoja mažiau kodo eilučių.

Pirmasis tvirtinimas yra subjektyvus. Dėl antrojo – mūsų pirmoji programa turėjo 58 PHP skripto eilutes, tuo tarpu kai `holiday.php` teturi 39 linijas, tai beveik trečdaliu mažiau. Pažiurėkime į programą, kaip tai buvo padaryta. Pirmosios tris linijos yra identiškos ir nereikalauja paaiškinimo:

```
$Price=500;
$StarModifier=1;
$CityModifier=1;
```

Ketvirtoji linija pristato naująjį kintamąjį, `$DestGrade`, kuris yra kintamųjų `$Destination` ir `$Grade` sąjunga.

```
$DestGrade = $Destination.$Grade;
```

Taigi, jei jūs pasirinksite Barseloną ir keturias žvaigždutes, `$DestGrade` bus `BarcelonaFour`. Atkreipkite dėmesį, kad mes tą patį galėjome padaryti ir prieš tai buvusiame pavyzdyje taip sutaupę keleta eilučių. Tačiau tai nebūtu labai didelis sumažinimas ir neleistu mums pademonstruoti gryno `if` elemento. Sumažinimas kurį mes čia padarėme yra jau žymus ir mes tik tikriname mūsų sąlygas su `$DestGrade`:

```
switch($DestGrade) {
```

Viskas ką mums reikia padaryti yra įsitikinti, kad mes perimame visus galimus variantus. Viskas kas netelpa į tris viešbutis padaugintus iš dviejų kainų turi būti neteisingai užpildyti užklausimai. Tai reiškia,k da mūsų visi variantai yra – `BarcelonaThree`, `BarcelonaFour`, `PragueThree`, `PragueFour`, `ViennaThree`, ir `ViennaFour`, taigi mums reikia perimti kiekvieną iš jų.

Vsi šie variantai atlieka panačius veiksmus, tad mes neaiškinsime kiekvieno iš jų. Pažiurėsime tik į vieną ir pperžvelgsime jį:

```
case "BarcelonaThree":
   $CityModifier=2;
   $Price = $Price * $CityModifier;
   echo "The cost for a week in $Destination is $Price";
   break;
```

`BarcelonaThree` atvėju, mes nustatėme `$CityModifier` reikšmę du ir padauginome ją iš kainos. Tada pateikėme atsakyma ekrane ir nutraukėme tolimesnį programos veikimą. Jei reikšmė `$DestGrade` nesutampa su jokia, tai reiškia, kad kažkas yra blogai ir mes siunčiame vartotą atgal:

```
default:
echo ("Go back and do it again");
break;
```

Laikantis gerų programavimo manierų mes gale rašome `break` gale, nors jis nedaro nieko. Paprasčiausiai mažiau tikėtina, kad jūsų kodas sukels problemų, net jei gale parašysite dar viena `case` po `else`. Viskas paprasta. Dabar kai apžvelgėme `if` ir susijusi `switch` elementus, padarysime praktinį pavyzdį.

Formos tikrinimas

Paskutiniame skyriuje mes minėjome, kad yra imanoma su-lugdyti mūsų paskolos programą atsitiktinai arba specialiai pateikiant klaidingą informacija. Pavyzdžiui, laukelyje kuris klausia amžiaus, galima parašyti „Ne tavo reikalas“. Kaip jūsų PHP programa išspręs šias problemas? Jūs galite tikėtis, kad jūsų vartotojai nerašys nesamonių, bet kiekvienu atvėju kai programa pateikiama viešumon, jūs būsite nustebę, dėl galimų atsakimų gausos, pradedans "999", ir baigiant žodine išraiška "keturesdešimt keturi", bet jūsų PHP programai tai visiškai beverčiai duomenys.

Ši problema sprendžiama uždraudžiant tam tikrus simbolius konkrečiuose laukeliuose. Mūsų pavyzdyje, mes perduodame vartotojo amžių kintamajam `$Age`. Tad PHP puslapyje mes galime patikrinti įvesta reikšmę su realistiškų skaičių eile:

```
if ($Age<1 or $Age>120)
  {
    echo "Incorrect Age value entered";
    break;
```

```
  }
```

ir tai užtikrins, kad duomenys daugiau mažiau teisingi.

Exit elementas

Jūs taip pat galite panaudoti kitą elementą vietoj `break,` – exit.

```
if ($Age<1 or $Age>120)
  {
    echo "Incorrect Age value entered";
    exit;
  }
```

Jei mes tikriname formą ir žinome, kad kažkas blogai įvedė duomenis, nėra jokio tikslo juos tikrinti ir atlikti kitus veiksmus toliau, taigi mes galime tuoj pat sustoti panaudoja komandą `exit`, kuri savo funkcijom yra tokia pati kaip ir mūsų jau žinomas `break`.

Daugiau nebus vygdoma nei HTML, nei PHP kodas, ar tekstas, jei `if` elemente bus įvykdyta `exit` komanda.

Tačiau `exit` komanda staiga nutraukia puslapio vygdymą ir tuo pačiu serveris nebepateikia jokių likusių simbolių, tame tarpe neuždaro ir jau atidarytu HTML tag'u, tad `exit` komandos naudojimas turėtu būti atsargus.

Grįžkime prie mūsų paskolos programos ir apsaugokime ją nuo vartotjų klaidų.

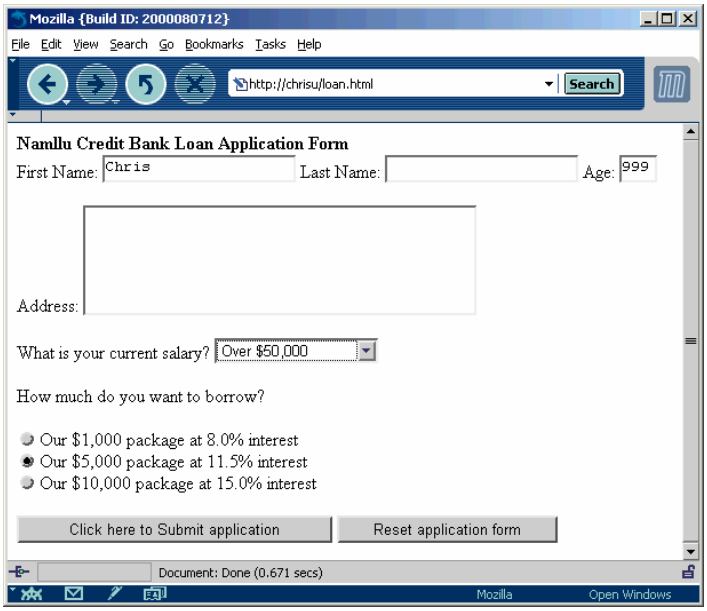
Išbandykite – formos tikrinimas

1. Grįžkite ir atidarykite `loan.php` ir pakeiskite sekanti kodą su jau egzistuojantčiu:

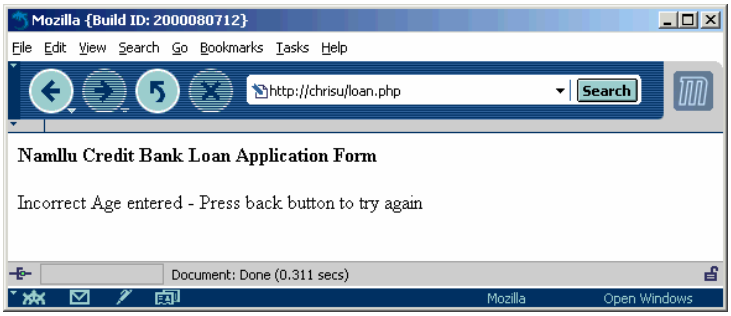
```
  <HTML>
  <HEAD></HEAD>
  <BODY>
  <B>Namllu Credit Bank Loan Application Form</B>
  <BR>
  <BR>
  <?php
  if ($Age<10 OR $Age>140)
  {
  echo "Incorrect Age entered - Press back button to try
again";
  exit;
  }
  if ($FirstName=="" or $LastName=="")
  {
  echo "You must enter your name - Press back button to try
again";
  exit;
  }
  if ($Address=="")
  {
  echo "You must enter your address - Press back button to
try again";
  exit;
  }
  If ($Loan!=1000 and $Loan!=5000 and $Loan!=10000)
  {
  echo "You must enter a loan value - Press back button to
try again";
  exit;
  }
  $SalaryAllowance = $Salary/5;
  $AgeAllowance = ($Age/10 - ($Age%10)/10)-1;
  ...
```

2. Išsaugokite kaip `loan.php`.

3. Dabar grįžkite atgal ir pabandykite pasiųsti forma neįvesdami visos informacijos arba aiškiai neteisingą amžių.



4. Jūs turėtumėte gauti pranešimą apie klaidą:



Kaip tai veikia

Žinoma jūs bet kad galite įvesti kie nors kito adresą, arba ne savo amžių. Tačiau PHP to niekaip negali patikrinti. Mūsų naujasis kodas paprasčiausiai patikrina ar vartotojas netyčia nepamiršo įvesti duomenis, ir ar specialiai neįvedė aiškiai klaidingų savo metų. Pirma patikrinama ar amžius yra tarp 10 ir 140, kitu atvėju mes galime būti tikri, kad asmuo meluoja:

```
if ($Age<10 OR $Age>140)
  {
    echo "Incorrect Age entered - Press back button to try
again";
    exit;
  }
```

Mes parodome atitinkamą pranešimą ir baigiame vygdymą. Mums nereikia daryti nieko daugiau jei ši sąlyga nepatenkinta.

Antras `if` elementas patikrina ar įvestas vardas ir pavardė. Eilutė "" yra tuščios eilutės kintamasis ir štai tai mes tikriname:

```
  if ($FirstName=="" or $LastName=="")
    {
      echo "You must enter your name - Press back button to
try again";
      exit;
    }
```

Ta patį padarome ir su adresu, patikrindami ar kintamasis `$Address` nėra tuščias:

```
  if ($Address=="")
    {
      echo "You must enter your address - Press back button
to try again";
      exit;
    }
```

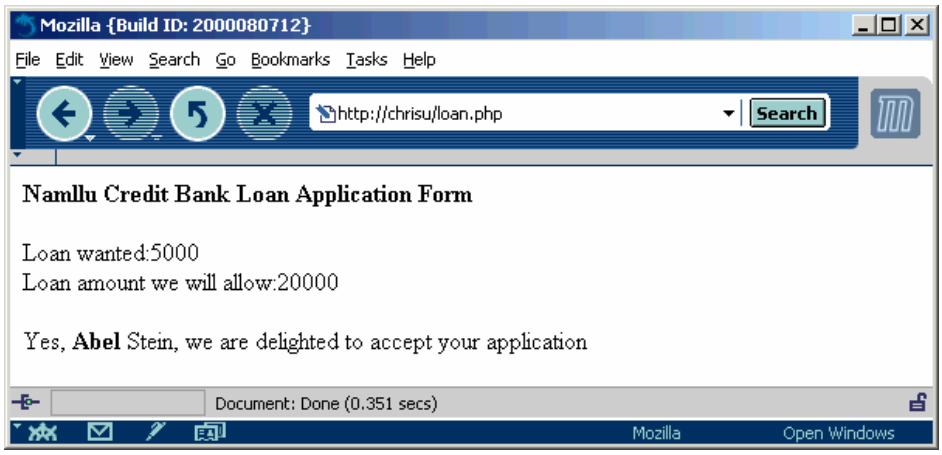
Galiausiai, mes patikriname pasirinkimo mygtukus vienai iš trijų galimų reikšmių:

```
  If ($Loan!=1000 and $Loan!=5000 and $Loan!=10000)
    {
      echo "You must enter a loan value - Press back button
to try again";
      exit;
    }
```

Jei paskolos suma nelygi vienai iš nustatytu reikšmių mes žinome, kad vartotojas nepasirinko reikšmės.

Piktybiškas skriptinimas – HTMLSpecialChars

Tai visiškai neišsprendė mūsų kodo patikimumo – jei jūs teisingai užpildėte kodą, bet parašėte sekanti vardą `<B>Abel</B>`, turėtumėte pamatyti štai tai:



Jūs galite pagalvoti, kad neatsitiko nieko neįprasto išskyrus vardą Abel. Jis paryškintas. Taip atsitiko todėl, kad naršyklė interpretavo HTML tag'ą <B>. Dabar įsivaizduokite, kad kažkas specialiai parašė piktybinį HTML kodą, arba kas dar pavojingiau skriptą. Tai gali priversti programą dirbti jiems.

Laimei PHP pateikia gerą funkciją sustabdančia tokius dalykus - `HTMLSpecialChars`. Tam tereikia eilutės tipo argumento:

```
  $String = HTMLSpecialChars("<B>This won't display the Bold
tags </B>");
```

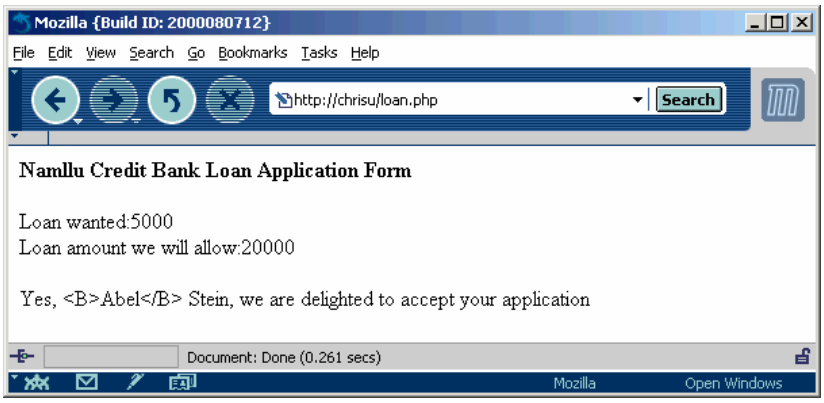
arba kintamojo vardo:

```
$String ="<B>This won't display the Bold tags </B>";
$String = HTMLSpecialChars($String);
```

Ši funkcija paverčia bet kokį HTML tag'ą į paprasta teksta kurį mes norime pateikti, šie tag'ai nevygdomi naršyklės. Taigi jei norime tai sustabdyti turime kiekvienam teksto laukelio reikšmę paleisti per žemiau pateiktą funkciją:

```
$FirstName = HTMLSpecialChars($FirstName);
$LastName = HTMLSpecialChars($LastName);
$Address = HTMLSpecialChars($Address);
$Age= HTMLSpecialChars($Age);
```

Mes turime pateikti šį kodą kur nors failo pradžioje ir vietoj HTML bus patekta



Dabar mūsų programa kiek saugesnė.

Apibendrinimas

Šiame skyriuje mes pristatėme pagrindinę PHP kalbos dalį – sprendimų priėmimą. PHP kalboje sprendimų pagrindas yra `if` elementas. Mes taip pat turėjome galimybę susipažinti su Loginiais operatoriais – daugiaus nei, mažiaus nei, lygybės, neligyvės, `AND`, `OR`, ir `NOT`. mes pamatėme, kad `if` elementas turi keletą skirtingų formatų ir yra gana lankstus apdorojant daugybinius sprendimus ir jis gali būti praplečiamas `elseif`. `Switch` elementas pasiūlė mums dar geresnį sprendimų apdorojimą, tačiau kai kyla klausimas – ką vartoti, `if`, `elseif`, ar `switch`, tai priklauso tik nuo programuotojo pasirinkimo.

Mes pabaigėme šį skyrių pagerindami mūsų paskolų programą, pasinaudodami naujai išmoktom galimybėm. Tačiau šios technikos bus naudojamos ir toliau visoje knygoje ir tai yra labai svarbu. Kaip jūs tikriausiai pastebėjote, pavyzdžiai vis didėja, o kodas juose kartojasi. Kitame skyriuje mes susipažinsime kaip PHP elgiasi su pasikartojimais pasinaudodama ciklais. Pagrindinis ciklų vartojimo privalumas, tas, kad jie sumažina naudojamo kodo apimtį. Jie taip pat gali būti panaudoti rašant kintamuosius į masyvus daug greičiau ir efektyviau, nei linijinis kodavimo būdas.

5
Ciklai ir Masyvai

Mes pristatėme fundamentalius programavimo pagrindus prieš tai buvusiame skyriuje – sprendimų priėmimą. Mes sužinojome, kad kodas gali būti vykdomas net tik paeiliui, bet ir priklausomai nuo tam tikrų programuotojo sukurtų sąlygų. Šiame skyriuje mes pristatysime tai ką kompiuteris daro geriausiai ir tiesa pasakius tam jie ir buvo sukurti – pasikartojančių užduočių vykdymas. Jei jūs turite atlikti tą pačią užduoti kas dieną, kas valanda, vėl ir vėl, jūs anksčiau ar vėliau įvesite klaidą. Panašią užduotį paskyrus kompiuteriui, jei jis pirmą kartą atliks ją teisingai – teisingai ji bus vykdoma ir visais kitais atvejais.

Mechanizmas kurį naudoja dauguma programavimo kalbų, tame tarpe ir PHP, atliekant pasikartojančias užduotis yra **ciklai** (loops). PHP kalboje yra trijų rūšių ciklai, ir mes ruošiamės pusę skyrius skirti juos visus nagrinėjant. Kai jau baigsime su ciklais, persikelsime prie susijusių elementų – masyvų. Mes jau šiek tiek minėjome masyvus anksčiau šioje knygoje. Masyvas yra eilė suindeksuotu kintamųjų, kurie kartu su ciklais gali būti labai naudingi. Ciklai suderinti su masyvais gali sukurti šimtus, net tūkstančius kintamųjų, panaudojus tik tris ar keturias kodo eilutes. Kaip jau tapo įprasta šį skyrių pabaigsime praktiniu pavyzdžiu kuris turės tiek ciklus tiek ir masyvus.

Šiame skyriuje aptarsime šias temas pagal eilę

- `While` ciklas
- `Do while` ciklas
- `For` ciklas
- Masyvų kūrimas
- Duomenų gavimas iš masyvų
- Kaip masyvai yra indeksuojami
- Kaip masyvai gali būti saugomi
- Daugybiniai masyvai
- Praktinis ciklų ir masyvu pavyzdys

Ciklai

Prieš tai buvusiame skyriuje mes pristatėme šakojimosi (pasirinkimo) elementus, kurie leido pristatyti sprendimų priėmimą PHP kode. Ciklai kažkuo panašūs į šakojimąsi, nes abiejuose atvejuose priklausomai nuo sąlygos sekanti kodo eilutė gali būti vykdoma arba ne.

Tačiau ciklai skiriasi nuo palyginimo operatorių nes tas pats veiksmas gali būti kartojamas daugybe kartų. Sąlyga yra patikrinama ir jei ji teigiamas kodas cikle yra vykdomas. Tada vėl patikrinama sąlyga; jei vėl teigiama ciklas atliekamas dar kartą; jei ji vis dar teigiama, kodas dar kartą vykdomas ir taip daugybę kartų. Manau supratot esmę.

Tiesa pasakius ciklai turi dar šį tą bendro su sąlygos operatoriais, mes matėme kad pasirinkimo operaciją galime atlikti trimis būdais, pasirinkdami tą kuris mums konkrečiu atveju geriausiai tinka. Panašiai, yra trys ciklų tipai, kiekvienas kuris tina skirtingai situacijai. Dabar mes išanalizuosime kiekvieną atskirai.

while ciklas

Mes pradėsime `while` ciklą, nes jis pats paprasčiausias iš trijų, ir kiek panašus į `if` elementą. Kaip ir `if` elementas, jis tikrina sąlygos rezultatą. Priklausomai ar sąlyga buvo patenkinta ar ne, kodo dalis esanti tarp skliaustu yra vykdoma.

```
while (sąlyga yra teigiama)
 {
  vykdomas kodas tarp skliaustų
 }
```

Kai ciklo kodas yra įvykdomas, sąlyga viršuje yra tikrinama vėl ir kodas gali būti vykdomas vėl jei ji teigiama (true). Jei sąlyga yra patikrinama ir gaunamas neigiamas rezultatas, kodas tarp skliaustu bus ignoruojamas ir PHP vykdys pirmąją eilutę sekančia po riestinių skliaustelių. Pažiūrėkime į šita pseudo-kodo pavyzdį:

```
while (mėnulis yra pilnatyje)
{
vilkai staugs
}
```

Taigi jei mėnulis nėra pilnatyje, vilkai ne staugs, bet kol pilnatis vilkai staugs. Pažiūrėkime į kitą pavyzdį. Tarkim mes norime informuoti vartotoją apsiperkanti e-parduotuvėje, kad jo kredito limitas buvo viršytas. Mes galime panaudoti tokį kodą norėdami pranešti vartotojui apie kredito limito viršijimą:

```
while ($VisoPirkiniu > $KreditoLimitas)
  {
    echo ("Jūs viršijote savo kredito limitą, tad
paskutinis pirkinys bus pašalintas iš jūsų užsakymo sąrašo");
    $VisoPirkiniu = $VisoPirkiniu - $PaskutinisPirkinys;
    $PaskutinisPirkinys = $PriespaskutinisPirkinys;
  }
```

Taigi jei vartotojas viršija kredito limitą `$KreditoLimitas`, mes atšaukiame paskutini pirkinį (`$PaskutinisPirkinys)`, pašalindami jo reikšmę iš visos pirkinių kainos sumos (`$ShoppingTotal`). Tada mes pakaičiame kintamojo `$PaskutinisPirkinys` reikšmę į prieš paskutinio pirkinio reikšmę. Šiuo būdu mes galime kartoti ciklą, pašalindami viena po kito pirkinius, kol `$VisoPirkiniu` yra mažesnis nei `$KreditoLimitas`.

Jūs gal būt pastebėjote, kad šis ciklas gali tęsti amžinai jei `$KreditoLimitas` būtu pateiktas neigiamas skaičius. Jei jūs naudojate ciklą kurio reikšmė visada gali būtu true, gali susidaryti begalinis ciklas, neleisiantis jūsų programai baigti darbo. Tai nesukels jokių pranešimų apie klaidą, paprasčiausiai ciklas bus vykdomas vėl ir vėl. Kai rašote kodą kuriame bus ciklų turėkite tai galvoje.

do while ciklai

`do while` ciklas yra panašus į `while` cilką, išskyrus vieną aspektą: sąlygos operatorius yra tikrinamas ciklo pabaigoje, o ne pradžioje. Tai turi iš pirmo žvilgsnio, bet iš tikrųjų svarbų skirtumą – ciklas bus atliktas bent vieną kartą, net jei patikrinta gale sąlyga bus neteisinga.

```
do
 {
  vykdomas kodas tarp skliaustų
 }
while(sąlyga yra teigiama); – grįžtama ir atliekama
iš naujo
```

Taigi jūs turite naudoti `do while` jei norite, kad ciklas būtu atliekamas bent kartą, net jei sąlyga yra false. Tai svarbu. Jei mes grįšime prie mūsų pirkinių kodo ir pakeisime ciklą į `do while`, tai paveiks visą mūsų kodo veikimą:

```
do
 {
   echo ("Jūs viršijote savo kredito limitą, tad paskutinis
pirkinys bus pašalintas iš jūsų užsakymo sąrašo");
   $VisoPirkiniu = $VisoPirkiniu - $PaskutinisPirkinys;
   $PaskutinisPirkinys = $PriespaskutinisPirkinys;
 } while ($VisoPirkiniu > $KreditoLimitas);
```

Dabar mes parašome įspėjantį užrašą net gi prieš patikrindami ar pirkėjas tikrai viršijo savo kredito limitą. Tai tikrai ne tai ko mes norime!

Pažiurėkime kokioje situacijoje `do while` ciklas gali būti naudojamas. Jei mes keliautumėme autostrada nuo 1 posūkio iki 10 tai galima būtu parašyti taip:

```
do
  {
   Vauruojam iki kito posūkio;
  } while ($posūkis != 10);
```

Žiūrint į kodą matosi, kad norint pasukti iš autostrados, mes turime privažiuoti bent jau pirmąjį posūkį; taigi mums būtinai reikia bent kartą atlikti ciklą. Kitas geras PHP pavyzdys būtu tada jei jums tikrai reikia atlikti skaičiavimus bent kartą, tačiau tų skaičiavimų gali prireikti ir daugiau, kol gaunamas norimas rezultatas. Pavyzdžiui mes turime pirminio skaičiaus paieškos programą. Norint patikrinti ar skaičius yra pirminis, mes turime jį dalinti iš visų skaičių pradedant dviems ir baigiant jo pačio reikšme minus vienas. Tai lengvai gali atlikti `while` loop cilkas:

```
do
 {
  $Remainder = $PossiblePrimeNumber%$Number;
  $Number=$Number+1;
 } while ($Remainder!=0 AND $Number<$PossiblePrimeNumber);
```

Čia mes dalinome mūsų tikrinamą skaičių `$PossiblePrimeNumber` iš visų skaičių pradedant 2 ir baigiant vienu mažiau nei pats `$PossiblePrimeNumber`, naudodamiesi modulus operatoriumi nustatydami ar yra dalybos likutis. Per kiekvieną ciklą mes atliekame dalybą ir patikriname jos likutį. Jei nerandamas joks likutis per visą ciklą, tai reiškia, kad skaičius negali būti pirminis.

Mes turime atlikti bent vieną ciklą, norėdami sužinoti ar mūsų kandidatas dalinasi iš kito skaičiaus taigi `do while` čia labiausiai tinkamas. Jei mes praeisime visą ciklą su duotuoju skaičiumi neaptikdami nei vienos dalybos, žinosime kad mūsų skaičius dalinasi tik iš 1 ir savęs pačio – taigi jis pirminis.

Kita dažna situacija kur `do while` yra labai naudingas yra tada kai mums reikia laukti vartotojo reakcijos. Pavyzdžiui, ar atsimenate mūsų sukurta spėjimo žaidimą 4 Skyriuje, kur PHP generuodavo atsitiktinį skaičių nuo 1 iki 10? Kas bus jei mes norėsime taip pakeisti kodą, kad vartotojas spėliotu tol kol jo pasirinktas skaičius bus teisingas? Paprasčiausias sprendimas yra į programą įtraukti `do while` ciklą, kuris būtų vykdomas, kol atspėjamas teisingas skaičius.

Dabar sustiprinkime mūsų `do while` ciklo pažinimą pavydžiu. Su jau anksčiau pateiktu pirminio skaičiaus pavyzdžiu mes sukursime pilnai veikiančią PHP programą.

Išbandykite – do while naudojimas

1. Su savo teksto redagavimo programa sukurkite tokį failą:

```
<HTML>
 <HEAD></HEAD>
<BODY>
<FORM METHOD=POST ACTION="check.php">
What is your Number:
<INPUT NAME="Guess" TYPE="Text">
<BR>
<BR>
<INPUT TYPE=SUBMIT VALUE="Click here to check if it is
prime">
<BR>
</FORM>
</BODY>
</HTML>
```

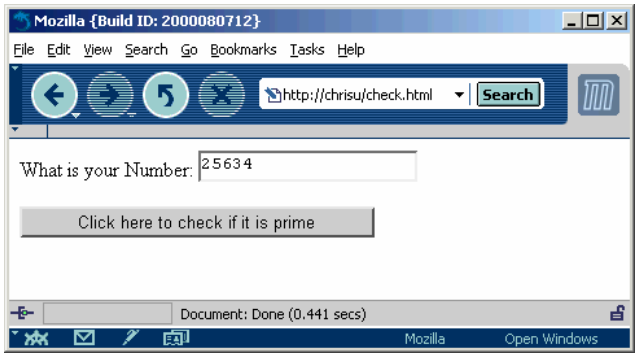
2. Išsaugokite kaip `check.html`.

3. Atsidarykite naująjį langą ir parašykite sekantį kodą:

```
<HTML>
<HEAD></HEAD>
<BODY>
<?php
$Count=2;
do
  {
    $Remainder = $Guess%$Count;
    $Count=$Count+1;
  } while ($Remainder!=0 AND $Count<$Guess);
if (($Count<$Guess) || ($Guess==0))
 {
  echo ("Your number is not prime");
 }
else
 {
   echo ("Your number is prime");
 }
?>
</BODY>
</HTML>
```

4. Išsaugokite kaip `check.php`.

5. Atidarykite `check.html` savo naršyklėje ir įveskite skaičių:



6. Paspauskite ant mygtuko norėdami sužinoti atsakymą:

Kaip tai veikia

`check.html` vienintelis tikslas yra gauti skaičių iš vartotojo:

```
What is your Number:
<INPUT NAME="Guess" TYPE="Text">
```

Šis skaičius yra perduodamas `$Guess` kintamajam kuris naudojamas `check.php` faile. Čia mes visų pirma nustatome skaičiavimo kintamojo reikšmę:

```
$Count=2;
```

Jis yra 2, mes nepradedame nuo 1 nes šiaip ar taip bet koks skaičius dalinasi iš 1. toliau prasideda ciklas:

```
do
{
```

Ciklo viduje mes tikriname liekaną likusią po `$Guess` ir `$Count` dalybos. Tada mes didiname mūsų `$Count` kintamojo reikšmę:

```
$Remainder = $Guess%$Count;
$Count=$Count+1;
```

Šios dvi linijos ir sudaro visą ciklo pagrindą. Savo pavyzdyje mes įvedėme 25634, tad šis ciklas dalino skaičių 25634 iš 2, 3, 4, 5, 6, 7, 8, ir taip iki pat 25633. Tai didelis kiekis operacijų kurias turi atlikti mūsų ciklas. Kiekvieno ciklo pabaigoje mes patikriname ar nepatenkintos dvi sąlygos: ar nėra dalybos liekanos ir ar kintamasis `$Count` nepasiekė tiriamo skaičiaus reikšmės:

```
} while ($Remainder!=0 AND $Count<$Guess);
```

Savo ciklą mes galime užbaigti tik tada kai bent viena iš salygų buvo patenkinta. Jei ciklas užsibaigė, bet `$Count` nesutampa su `$Guess`, reiškias tikrinamas skaičius buvo padalintas iš skaičiaus esančio `$Count` kintamajame, tad `$Guess` nėra pirminis. Taip pat mes turime prisiminti apie 0 iš kurio dalinant visi atsakymai bus lygus nuliui, tačiau pats 0 nėra pirminis. Iš kitos pusės jei du kintamieji sutampa, reiškias vartotojo pateiktas skaičius yra pirminis.

```
if (($Count<$Guess) || ($Guess==0))
 {
  echo ("Your number is not prime");
 }
else
 {
  echo ("Your number is prime");
 }
```

Paskutinis būtinas veiksmas yra pateikti vartotojui atsakymą. Dabar laikas susipažinti su paskutiniuoju ciklų tipu.

for Loops

`for` geriausiai naudoti kai ciklas turi būti vykdomas iš anksto nustatytą kartų skaičių. Tariant kitais žodžiais jis suteikia jums galimybę nurodyki kiek kartų ciklas turi būti vykdomas. Sąlygos dalis yra kiek sudėtingesnė nei `while` ciklo, nes susideda iš trijų dalių:

```
for (ciklo skaitiklis; ciklo skaitiklio patikrinimas;
skaitiklio keitimas)
{
vykdomas kodas tarp šių eilučių
}
```

Kartu su `for` ciklo mes pristatome ir naują koncepciją – ciklo skaitiklis. Tai kintamasis skirtas apkaičioti kiek kartų reikia atlikti ciklą ir jį užbaigti kai šis skaičius viršijamas. Trečioji sąlygos dalis užtikrina, kad mes pakeičiame skaitiklio reikšmę kiekviename cikle. Šios trys dalus užtikrina, kad mes galime sukonstruoti sudėtingas sąlygas ir ciklų konstrukcijas. Tačiau iš tikro nei viena ši dalis nėra privaloma, tačiau norėdami geriau suprasti ciklo veikimo principus mes naudosime jas visas.

Pažiūrėkime kaip `for` ciklas gali būti panaudotas. Pirmasis pavyzdys: atspausdinsime mūsų vardą 10 kartų stulpeliu. Žinoma mes galine pasinaudoti `echo` komanda dešimt kartų, tačiau tai būtu nelabai racionalu. Taip pat mes galėtume naudoti ką tik išmoktą `while` ciklą, ir sukurti skaičiuojanti kintamąjį ciklo viduje, kaip pateikta žemiau:

```
$counter=0;
while ($counter<10)
 {
   echo "My name is Chris!";
   $counter=$counter+1;
 }
```

Tačiau kiek kartu būtu atliekamas šitas ciklas? Ar tai 9 nes kai pasiekiame 10 ciklas sustoja? Ar tai 11 nes ciklo sąlyga pradedame skaičiuoti nuo 0? Iš tikro jis bus atliktas 10 kartų, tačiau mes turime atlikti daug daugiau loginių apmąstymų, kad įsitikinti jog viskas gerai. Ciklo pasikartojimo skaičių geriausiai skaičiuoti su `for` ciklu:

```
for ($counter=1; $counter<=10; $counter++)
 {
  echo "My name is Chris!";
 }
```

Ciklo skaitiklis gali būti panaudotas ir pačio ciklo viduje:

```
for ($counter=1; $counter<=10; $counter++)
 {
  echo $counter;
 }
```

Tai pateiktu tokią išraišką 12345678910. Jūs tikriausiai pastebėjote, kad mes pristatėme naują terminą kurio nenaudojome anksčiau: didinimo operatorių ++. Jis tapatus eilutei:

```
$counter=$counter+1;
```

Vėlgi tai tik patogus sutrumpinimas.

Pabandykime padaryti pavyzdį kuris sukurtu dinaminę formą. Jis paima reikšmes iš vartotojo ir jomis remiantis sukuria HTML elementus sekančiame puslapyje. Tada elementai pateikiami vartotojui trečiajame puslapyje. Pabandykite, tai nėra taip sudėtinga.

Išbandykite – for ciklo naudojimas

1. Neleiskite ilsėtis savo teksto redaktoriui ir parašykite:

```
<HTML>
<HEAD></HEAD>
<BODY>
<FORM METHOD=POST ACTION="dynamic.php">
How many children do you have?
<INPUT NAME="Number" TYPE="Text">
<BR>
<BR>
<INPUT TYPE=SUBMIT>
<BR>
</FORM>
</BODY>
</HTML>
```

2. Išsaugokite kaip `dynamic.html`.

3. Uždarykite šį failą, ir sukurkite naująjį:

```
<HTML>
<HEAD></HEAD>
<BODY>
<FORM METHOD=GET ACTION="dynamic2.php">
<?php
for ($Counter=0; $Counter<$Number; $Counter++)
{
$Offset = $Counter+1;
echo "<BR><BR>Please enter the name of child number
$Offset<BR>";
echo "<INPUT NAME=Child[] TYPE=TEXT>";
}
if ($Counter==0) echo"Press the button to move on";
?>
<BR>
<BR>
<INPUT TYPE=SUBMIT>
</FORM>
</BODY>
</HTML>
```

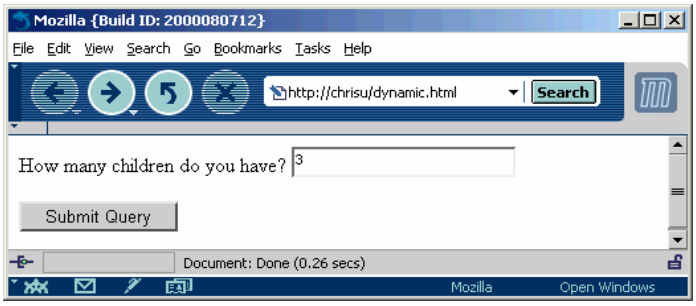
4. Išsaugokite kaip `dynamic.php`.

5. Vėl gi uždarykite šitą failą ir sukurkite dar vieną:

```
<HTML>
<HEAD></HEAD>
<BODY>
<?php
$Count=0;
echo "Your children's names are:";
do
{
echo"<BR><BR>$Child[$Count]";
$CheckEmpty = "$Child[$Count]";
$Count=$Count+1;
} while ($CheckEmpty!="");
if ($Count==1) echo "Not Applicable";
?>
</BODY>
</HTML>
```

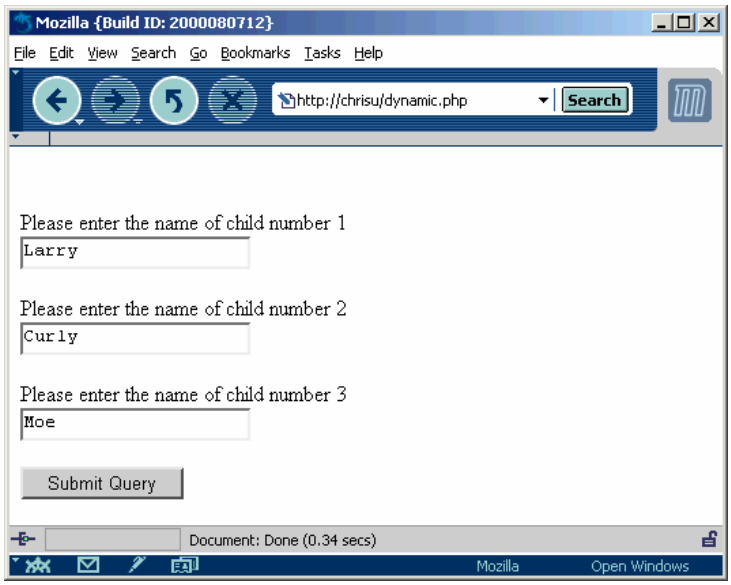
6. Išsaugokite kaip `dynamic2.php`.

7.Uždarykite jį. Tada atidarykite `dynamic.html` savo naršyklėje ir parašykit skaičių didesni nei 0, net jei neturite vaikų:

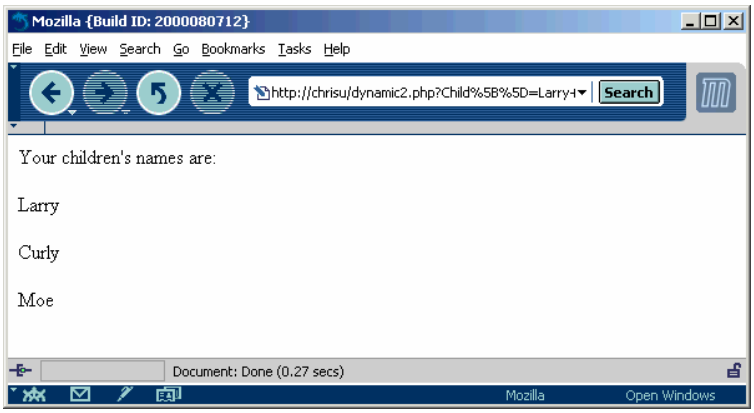


8. Paspauskite ant Submit Query ir sekančiame puslapyje pamatysite tiek eilučių kokį skaičių parašėte:

9. Parašykite kokius nors vardus:



10. Paspauskite ant Submit Query iš jūsų įvesti vardai bus pateikti.



Kaip tai veikia

Ši paprasta užduotis nėra tokia paprasta kaip gali pasirodyti iš pradžių. Mes pradedame paprastu HTML failu `dynamic.html`, kuris paprašo įvesti vaikų skaičių. Šis skaičius perduodamas į kintamąjį `$Number dynamic.php` skripte.

```
How many children do you have?
<INPUT NAME="Number" TYPE="Text">
```

Tada jau mūsų PHP programoje mes padarome nedidelį triuką. 1 Skyriuje mes jau minėjome, kad pasiuntus HTML tag'us kaip tekstą PHP skripte jis paverčiamas į HTML. Tas pats principas panaudotas ir čia. Taigi siųsdami `<FORM>` tag'ą kaip `echo` komandą, mes nurodome naršyklei, kad norime sukurti formą.

```
<FORM METHOD=GET ACTION="dynamic2.php">
<?php
echo
```

Kai jau esame `<FORM>` tag'u viduje mums reikia sukurti norimą teksto langelių skaičių. Mes žinome kiek jų mums reikia, kadangi vartotojas jau įvedė šį skaičių prieš tai buvusiame puslapyje. Šis skaičius saugomas kintamajame `$Number`.

Pats laikas pristatyti ir patį `for` ciklą. Mes norime parodyti tiek teksto langeliu, kiek yra `$Number` reikšmė, tačiau jei vartotojas neturi nei vieno vaiko mes neturime parodyti nei vieno tokio langelio. Tai reiškia, kad mes trume pradėti `$Counter` nuo nulio, ir jei kintamojo `$Number` reikšmė yra 0, mes iš viso praleisime ciklą jo neatlikdami. Tad ciklo skaitiklis pradeda skaičiuoti nuo nulio ir didėja vienu vienetu kiekvieną ciklą. Ciklas kartojamas tol kol skaitiklio reikšmė mažesnė nei vaikų skaičius:

```
for ($Counter=0; $Counter<$Number; $Counter++)
{
```

Ciklo viduje mes susiduriame su kita problema. Mes turime personalizuoti kiekvieną teksto langelį juos sunumeruojant. Tačiau jei mes naudosime skaitiklį jo reikšmė bus vienetu mažesnė nei turėtu būti. Tad mes turime "apeiti" `$Counter`, įvesdami dar vieną kintamąjį, kurio reikšmė būtu `$counter` plius vienas:

```
$Offset = $Counter+1;
```

Mūsų programa jau įgauna aiškius bruožus. Mes parodome pranešimą kuriame paprašome įvesti vaiko vardą:

```
echo "<BR><BR>Please enter the name of child number
$Offset<BR>";
```

Tada pateikiame teksto langelį. Pastebėkite, kad nurodydami teksto eilutės vardo atributą, mes parašėme laužtinius skliaustus po jo, taip nurodydami, kad naujas kintamasis yra masyvas. Kiekvieną kartą kai vardas perduodamas kintamajam `$Child`, jis bus išsaugotas naujame elemente: `$Child[0]`, `$Child[1]`, `$Child[2]` ir taip toliau:

```
echo "<INPUT NAME=Child[] TYPE=TEXT>";
}
```

Ciklas čia ir užsibaigia. Tad viskas ką mes padarėme yra kiekviena karta vykdydami ciklą pateikiame truputėli teksto ir teksto laukelį. Po to kai ciklas užbaigtas, mes einame toliau ir parašome Submit mygtuką, ir uždarome pačią formą. Tačiau prieš tai padarant mums reikia patikrinti ar vartotojas neįvedė nulio. Jei taip ir atsitiko, mes parodome kitą pranešimą:

```
if ($Counter==0) echo"Press the button to move on";
?>
```

Trečioji programa paprasčiausiai atspausdina vardus kurie saugomi masyve. Skamba gana paprastai, tačiau visų pirma jūs turite turėti omenyje, kad kintamasis `$Number` nebuvo perduotas naujajam skriptui. Jis egzistavo tik paskutiniame puslapyje – `dynamic.php`. Taigi, šioje situacijoje mes numanome, kad masyve `$Child[]` yra įvesti vaikų vardai, tačiau nežinome kiek jų yra. Tai reiškia, kad mes turime pasinaudoti mūsų senu draugu `do while` ciklu ir apskaičiuoti įvestų vardų skaičių rankiniu būdu. Mes pradėsime nuo nulio, nes ir vardų gali būti nulis.

```
$Count=0;
```

Mes parodome tekstą web puslapyje:

```
echo "Your children's names are:";
```

Tada pradedame `do while` ciklą, nes norime, kad ciklas būtu įvykdytas bent kartą:

```
do
 {
```

Pradėdami ciklą mes pateikiame pirmąjį masyvo elementą, `$Child[0]`. Jei jis tuščias, tada nieko išskyrus tuščią eilutę nebus parodyta. Kitu atveju bus pateiktas pirmojo vaiko vardas:

```
echo"<BR><BR>$Child[$Count]";
```

Tada kintamajam `$CheckEmpty` mes priskiriame `$Child[0]` reikšmę. Jei ji tuščia, `$CheckEmpty` sukels ciklo pabaigą. Jei ne, mes vykdysime ciklą toliau.

```
$CheckEmpty = "$Child[$Count]";
```

Mes padidinsime skaitiklį:

```
$Count=$Count+1;
```

Tada patikrinsime ar `$CheckEmpty` nėra tuščias. Jei taip ir yra mes užbaigsime savo ciklą; jei ne, tesiame:

```
} while ($CheckEmpty!="");
```

Galiausiai, mes turime patikrinti, gal įvestų vaikų skaičius buvo 0. Jei taip ir buvo mes pateikiame atitinkama pranešimą:

```
if ($Count==1) echo "Not Applicable";
```

Šitas pavyzdys ne tik pademonstravo for ciklo naudojimą, bet ir pristatė mūsų sekančią temą – masyvus. Priežastis, kodėl mes tik dabar pradėjome apie jas pokalbį yra ta, kad masyvų naudojimas labai palengvina darbą, tačiau be galimybių kurias mes dabar jau žinome, jais sunku pasinaudoti. Kas būtu jei jums reiktu įvesti 20 vaikų? Be ciklų jūsų kodas turėtu 20 eilučių nuskaityti vardam, ir dar 20, kad juos pateikti atgal. O dabar įsivaizduokite programas kurios dirba su tūkstančiais ir milijonais žodžių?

Be ciklų mes prarandame ir dalį dinamiškumo nes įvedimo laukelių skaičius būtu iš anksto nustatyta, nebūtu galima jų generuoti dinamiškai.

Masyvai

Mes jau trumpai pristatėme masyvus šioje knygoje, ir ką tik pasinaudojome jais vėl. Atėjo laikas apžvelgti juos išsamiau. Masyvai yra kintamųjų rinkinys turintis tą patį vardą, bet kiekvienas turi skirtinga **indeksą**. Kiekvienas masyvo narys vadinamas **elementu**. Jūs galite sukurti masyvus tokiu pat būdu kaip ir paprastus kintamuosius, tik nepamirškite parašyti laužtinių skliaustų kuriuose rašomas indeksas:

```
$StatesOfTheUSA[1] = "Washington";
$StatesOfTheUSA[2] = "California";
```

Jūs nebutinai turite priskirti reikšmes iš eilės, galite praleisti keletą ar per tiek kiek norite:

```
$StatesOfTheUSA[49]="Alaska";
$StatesOfTheUSA[13]="Alabama";
```

Tiesa pasakius jūsų galite visiškai atsisakyti skaitinės numeracijos ir naudoti raides. Tokie masyvai kaip šis dažnai vaidinami asociaciniais masyvais:

```
$StateCapital["ca"] = "Sacramento";
$StateCapital["il"] = "Springfield";
```

Atkreipkite dėmesį, kad norėdami pasiekti asociacinio masyvo elemento reikšmę, jūs galite nerašyti kabučių jei norite. Norint parašyti Sacremento, galite rašyti:

```
echo $StateCapital["ca"];
```

arba

```
echo $StateCapital[ca];
```

Abu pateiks tokius pačius atsakymus.

Dar vienas PHP masyvo privalumas tas, kad to pačio masyvo elementams galite priskirti skirtingų tipų duomenis. Štai pavyzdys:

```
$Number[1]=24;
$Number[2]="twenty three";
$Number[2]=$variable;
$Number["ca"]=$variable;
```

Tačiau čia kyla keletas klausimu. Kaip PHP žino kokio dydžio masyvas turi būti? Ir kiek atminties ji turi skirti masyvui?

Masyvų inicializacija

Pradinių masyvo reikšmių nustatymas vadinamas inicializacija. Mes jau susisūdėme su masyvų inicializacija du kartus šioje knygoje: mes nesirūpinome dėl indeksavimo ir leidome viską padaryti pačiai PHP. Mes sukuriame vieną masyvo elementą, tada sekantį tokiu pačiu vardu:

```
$Author[]="William Shakespeare";
$Author[]="Franz Kafka";
```

Be laužtinių skliaustų PHP nežinotų, kad mes operuojame masyvo elementais, ir pakaistu pirmąją reikšmę antrąją: laužtiniai skliaustai nurodo, kad mes norime saugoti duomenis masyve. Kadangi nėra indekso reikšmės PHP pati nusprendžia kokias suteikti. Jūs įsitikinsite, kad jei `$Author[]` masyvas nebuvo naudotas anksčiau, reikšmės pateiktos viršuje bus išsaugotos kaip `$Author[0]` ir `$Author[1]`. PHP pati priskirs reikalingą indeksą.

Mes taip pat jau susidūrėme ir su kitu būdu priskirti naują elementą masyvui nurodant elementų indeksus:

```
$Author[0]="William Shakespeare";
$Author[1]="Franz Kafka";
```

Šitame pavyzdyje mes patys nustatome masyvo elementų indeksus, neleisdami to padaryti PHP – indeksavimas gali vykti ir ne pagal eilę, kaip jau rašyta anksčiau. PHP skiriasi nuo daugelio programavimo kalbų dviem aspektais. Visų pirma mums nereikia nustatyti duomenų tipo saugomo masyve, tai susyja su visa PHP esme kai jums nereikia nustatinėti kintamųjų tipo nes PHP puikiai tai padaro pati. Antrasis išskirtinumas yra tas, kad mums nereikia nurodyti kokio dydžio bus masyvas prieš pradedant juo naudotis. Dar karta pati PHP nustato kokio dydžio turi būti maksimalus indeksas; pavyzdžiui `$Author[]` tereikia dviejų elementų.

Yra dar būdai PHP kalboje sukurti masyvams. Abiejuose panaudota `array ()` konstrukcija. Jei mes paimtumėme mūsų autorių pavyzdį, jie į masyvą gali būti įtraukti šiuo būdu:

```
$Author = array ("William Shakespeare", "Franz Kafka");
```

Dar kartą PHP automatiškai sugeneruos indekso reikšmes. Dar kartą indeksas prasidės nuo nulio, ir naujosios reikšmės bus patalpintos į mažiausias dar neužimtas indekso reikšmes. Jei jums reiktu pavaizduoti `$Author[1]` turinį, tai būtu Franz Kafka.

Tokio tipo masyvams nėra jokio dydžio apribojimo, jūs taip pat galite parašyti ir tokią eilutę:

```
$StatesOfTheUSA = array ("Alabama", "Alaska", "Arizona",
"Arkansas", "California", "Colorado", "Connecticut",
"Delaware", "Florida", "Georgia", "Hawaii", "Idaho",
"Illinois", "Indiana", "Iowa", "Kansas", "Kentucky",
"Louisiana", "Maine", "Maryland", "Massachusetts",
"Michigan", "Minnesota", "Mississippi", "Missouri",
"Montana", "Nebraska", "Nevada", "New Hampshire", "New
Jersey", "New Mexico", "New York", "North Carolina", "North
Dakota", "Ohio", "Oklahoma", "Oregon", "Pennsylvania", "Rhode
Island", "South Carolina", "South Dakota", "Tennessee",
"Texas", "Utah", "Vermont", "Virginia", "Washington", "West
Virginia", "Wisconsin", "Wyoming");
```

Vėl gi pirmoji valstija gaus indeksą – 0, o Wyoming turės 49 indeksą.

Tačiau tai yra kiek neintuityvu, jūs žinote, kad JAV yra 50 valstijų, tad paskutinės valstijos indeksas – 49 gali jūs kiek sutrikdyti. Norint to išvengti `array()` funkcija, leidžia jums pradėti indeksą nuo to skaičiaus kokio jūs norite. Tam naudojamas => operatorius:

```
$StatesOfTheUSA = array (1 => "Alabama", "Alaska", "Arizona",
"Arkansas",
"California",    "Colorado",    "Connecticut",    "Delaware",
"Florida", "Georgia",
"Hawaii", "Idaho", "Illinois", "Indiana", "Iowa", "Kansas",
"Kentucky",
"Louisiana",    "Maine",    "Maryland",    "Massachusetts",
"Michigan", "Minnesota",
"Mississippi", "Missouri", "Montana", "Nebraska", "Nevada",
"New Hampshire", "New Jersey", "New Mexico", "New York",
"North Carolina", "North Dakota", "Ohio", "Oklahoma",
"Oregon", "Pennsylvania", "Rhode Island", "South Carolina",
"South Dakota", "Tennessee", "Texas", "Utah", "Vermont",
"Virginia", "Washington", "West Virginia", "Wisconsin",
"Wyoming");
```

Tariant kitais žodžiais, jūs nurodote nuo kokio indekso pradėti ir parašote => operatorių, viskas kitas darosi pagal jau mums žinomas taisykles. Dabar jei jūs

parašytumėte `$StatesOfTheUSA[50]` turinį tai būtu Wyoming. Indeksas nebūtinai turi prasidėti 1, tai taip pat laisvai gali būti ir 101, tada Wyoming turėtu 150 indeksą.

Jei jūs norite indeksuoti didelį asociacinį masyvą, jums reikės kiekvieną elementą nusakyti individualiai. Pavyzdžiui tai gali būti padaryta štai taip:

```
$StatesOfTheUSA = array ("al" => "Alabama", "ak" => "Alaska",
"az" => "Arizona", "ar" => "Arkansas", "ca" => "California",
"co"  =>  "Colorado",  "ct"  =>  "Connecticut",  "de"  =>
"Delaware", "fl" => "Florida", "ga" => "Georgia", "hi" =>
"Hawaii", "id" => "Idaho", "il" => "Illinois", "in" =>
"Indiana", "ia" => "Iowa", "ks" => "Kansas", "ky" =>
"Kentucky", "la" => "Louisiana", "me" => "Maine", "md" =>
"Maryland", "ma" => "Massachusetts", "mi" => "Michigan", "mn"
=> "Minnesota", "ms" => "Mississippi", "mo" => "Missouri",
"mt" => "Montana", "ne" => "Nebraska", "nv" => "Nevada", "nh"
=> "New Hampshire", "nj" => "New Jersey", "nm" => "New
Mexico", "ny" => "New York", "nc" => "North Carolina", "nd"
=> "North Dakota", "oh" => "Ohio", "ok" => "Oklahoma", "or"
=> "Oregon", "pa" => "Pennsylvania", "ri" => "Rhode Island",
"sc" => "South Carolina", "sd" => "South Dakota", "tn" =>
"Tennessee", "tx" => "Texas", "ut" => "Utah", "vt" =>
"Vermont", "va" => "Virginia", "wa" => "Washington", "wv" =>
"West Virginia", "wi" => "Wisconsin", "wy" => "Wyoming");
```

Tai lėtas darymas, bet jis atlieka savo darbą.

Keliavimas per masyvą

Kai jau jūs sukūrėte didelį masyvą su daugybę įrašų, jums tikrai nesinorės grįžinėti ir individualiai gauti duomenis iš kiekvieno masyvo elemento. Tai labai didelis nereikalingas darbas. Štai čia masyvai ranka rankon dirba su ciklais. Jei jums reikia atspausdinti visų valstijų pavadinimus, `for` ciklas tai gali padaryti už mus. Norint pavaizduoti visas 50 valstijų masyve kurio indeksai yra nuo 1 iki 50 mums tereikės 3 kodo linijų, tame tarpe ir linijos kuri sukuria masyvą:

```
for ($counter=1; $counter<51; $counter++) {
echo"<BR>$StatesOfTheUSA[$counter]";
}
```

Linija tarp skliaustu parenka elementą iš `$StatesOfTheUSA` masyvo, parašant kintamąjį `$counter` tarp laužtinių skliaustu kaip masyvo indeksą. Kadangi `$counter` yra kintamasi, indeksas gali būti keičiamas keičiant `$counter` kintamojo reikšmę. Mūsų `for` ciklas sako: "Pradėkite skaičiuoti nuo 1 ir kilkite iki 50, didinant reikšmę vienu vienetu" Taigi `$counter` kintamasis gauna reikšmes 1, 2, 3, ir taip toliau:

```
echo"<BR>$StatesOfTheUSA[1];
echo"<BR>$StatesOfTheUSA[2];
echo"<BR>$StatesOfTheUSA[3];
t.t...
```

Niekas netrukdo mums naudotis `while` ar `do while` ciklus, nors jums tektu patiems sukurti ciklo skaitiklį ir priversti jį didėti. Sekantis `while` ciklas, darytu tokį patį veiksmą kaip ir `for` ciklas:

```
$counter=1;
while ($counter<51) {
echo"<BR> $StatesOfTheUSA[$counter]";
$counter=$counter+1;
}
```

Tereiktu parašyti keletą papildomų eilučių ir viskas.

Dabar padarykime nedidelį pavyzdį kaip judama per masyvą. Kadangi mes jau darėme tai anksčiau, mes kiek pasunkinsime jūsų užduotį. Šiame pavyzdyje mes sukursime du masyvus, vienas turės visų valstijų pavadinimus, kitas turės visų valstijų sostines. Jūs pasirinksite valstija iš sąrašo langelio ir programa ieškos atitinkančios tą valstiją sostinės. Mes panaudosime masyvus ir ciklus keletą kartų , norint užtikrint, kad mums nereikės sukurti 50 HTML linijų norint parodyti vieną atsakymą.

Išbandykite – judėjimas masyve

1. Atidarę savo teksto redaktorių parašykite:

```
<HTML>
<HEAD></HEAD>
<BODY>
<FORM ACTION="capitals.php" METHOD=POST>
What state do you want to know the capital of?
<SELECT NAME=State>
<?php
$StatesOfTheUSA = array (1 => "Alabama", "Alaska", "Arizona",
"Arkansas",
"California", "Colorado", "Connecticut", "Delaware",
"Florida", "Georgia",
"Hawaii", "Idaho", "Illinois", "Indiana", "Iowa", "Kansas",
"Kentucky",
"Louisiana", "Maine", "Maryland", "Massachusetts",
"Michigan", "Minnesota",
"Mississippi", "Missouri", "Montana", "Nebraska", "Nevada",
"New Hampshire", "NewJersey", "New Mexico", "New York",
"North Carolina", "North Dakota", "Ohio", "Oklahoma",
"Oregon", "Pennsylvania", "Rhode Island", "South Carolina",
"South Dakota", "Tennessee", "Texas", "Utah", "Vermont",
"Virginia", "Washington", "West Virginia", "Wisconsin",
"Wyoming");
for ($counter=1; $counter<51; $counter++) {
  echo"<OPTION>$StatesOfTheUSA[$counter]</OPTION>";
}
echo "</SELECT><BR><BR>";
for ($counter=1; $counter<51; $counter++) {
  echo"<INPUT TYPE=HIDDEN
NAME='HiddenState[]'VALUE='$StatesOfTheUSA[$counter]'>";
}
echo "<INPUT TYPE=SUBMIT></FORM>";
?>
</BODY>
</HTML>
```

2. Išsaugokite kaip `states.php`, nieko tokio, kad tai PHP, o ne HTML failas.

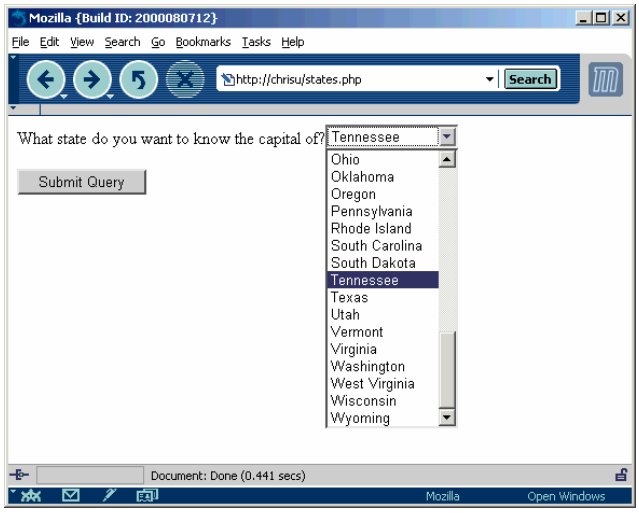
3. Atsidarę naują failą parašykite

```
<HTML>
<HEAD></HEAD>
<BODY>
<?php
$StateCapital = array (0 => "Montgomery", "Juneau",
"Phoenix", "Little Rock",
"Sacramento","Denver","Hartford", "Dover","Tallahasse",
"Atlanta", "Honolulu",
"Boise", "Springfield","Indianapolis", "Des Moines",
"Topeka", "Frankfort", "Baton
Rouge","Augusta","Annapolis","Boston", "Lansing", "Saint
Paul","Jackson",
"Jefferson City", "Helena","Lincoln", "Carson
City","Concord", "Trenton","Santa
Fe", "Albany", "Raleigh","Bismarck","Columbus","Oklahoma
City", "Salem",
"Harrisburg", "Providence", "Columbia","Pierre", "Nashville",
"Austin","Salt Lake
```

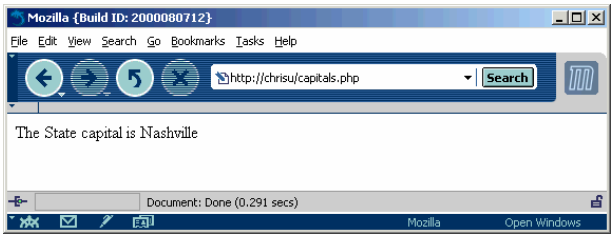
```
City", "Montpelier","Richmond","Olympia","Charleston",
"Madison","Cheyenne");
for ($counter=0; $counter<50; $counter++)
{
if($HiddenState[$counter]==$State)
{
echo"The State capital is $StateCapital[$counter]";
}
}
?>
</BODY>
</HTML>
```

4. Išsaugokite kaip `capitals.php`.

5. Atidarykite `states.php` ir pasirinkite valstiją:



6. Paspauskite ant Submit Query mygtuko, norėdami sužinoti atsakymą:



Kaip tai veikia

Šitas pavyzdys labai gerai pademonstruoja kiek kodo galima sutaupyti naudojant ciklus ir masyvus. Jums vien rašymo sumažėjo daug kartų. Šiame pavyzdyje mes taip pat pakeitėme pirmojo puslapio tipą į PHP, nes norėjome pasinaudoti PHP `array()` funkcija, kad nereikėtu spausdinti 50 HTML linijų `<SELECT>` sąrašo laukelyje. Mes pradėjome sukurdami HTML formą:

```
<FORM ACTION="capitals.php" METHOD=POST>
What state do you want to know the capital of?
```

Po to mes sukūrėme sąrašo laukelį:

```
<SELECT NAME=State>
```

Tada mes persikėlėme į PHP ir pradėjome pildyti valstijų pavadinimus į masyvą pavadintą `$StatesOfTheUSA`, pradėdami nuo indekso 1:

```
<?php
$StatesOfTheUSA = array (1 => "Alabama", "Alaska", "Arizona",
"Arkansas", "California", "Colorado", "Connecticut",
"Delaware", "Florida", "Georgia", "Hawaii", "Idaho",
"Illinois", "Indiana", "Iowa", "Kansas", "Kentucky",
  "Louisiana", "Maine", "Maryland", "Massachusetts",
"Michigan", "Minnesota", "Mississippi", "Missouri",
"Montana", "Nebraska", "Nevada", "New Hampshire", "New
Jersey", "New Mexico", "New York", "North Carolina", "North
Dakota", "Ohio", "Oklahoma", "Oregon", "Pennsylvania", "Rhode
Island", "South Carolina", "South Dakota", "Tennessee",
"Texas", "Utah", "Vermont", "Virginia", "Washington", "West
Virginia", "Wisconsin", "Wyoming");
```

Tada mes sukūrėme ciklas kuris kartojosi 50 kartų, patalpindami kiekvienos valstijos pavadinimą į sąrašo laukelį.

```
  for ($counter=1; $counter<51; $counter++) {
  echo"<OPTION>$StatesOfTheUSA[$counter]</OPTION>";
  }
  echo "</SELECT><BR><BR>";
```

Kai tik ciklas pasibaigė mes uždarėme sąrašo laukelį. Jūs gal būt tikėjotės, kad kita linija bus Submit mygtukas ir formos uždarymas. Tačiau, čia iškyla problema jei jūs galvosite iš karto ir apie sekantį puslapį. Visos valstijos mums reikalingos ir sekančiame puslapyje, ne tik šiame.

Kodėl? Pagalvokite kaip mes surasime konkrečios valstijos sostinę. Me sužinosime valstijos indeksą `$StatesOfTheUSA` masyve, ir tada ieškosime atitinkamo indekso sostinių masyve `$StateCapital`. Pavyzdžiui, `$StatesOfTheUSA[1]` turinys yra Alabama, taigi `$StateCapital[1]` turinys yra Montgomery. Mes galime atlikti tokius veiksmus, nes esama įsitikinę, kad indeksai abiejuose masyvuose sutampa savo reikšmėmis.

Dabar jei mes persiūsime tik tai pasirinktą valstiją į kitą skriptą, mes negalėsime nustatyti šios valstijos indekso nes neturėsime pačio masyvo. Be šios informacijos, mes negalėsime surasti ir atitinkamo indekso `$StatesOfTheUSA` masyve. Be šios informacijos mes negalėsime sužinoti ir atitikmens `$StateCapital` masyve.

Ką mums daryti? Mums tikrai reikia persiųsti duomenis apie pasirinktos valstijos indeksą į kitą puslapį. Vienas akivaizdus būdas yra persiųsti visą masyvą į sekantį skriptą. Taigi mes taip ir padarysime.

Mes sukuriame antrą ciklą, ir juo pasinaudodami sukuriame masyvą turinti 50 paslėptų laukelių, `HiddenState`. Mes užpildome juos valstijomis esančiomis `$StatesOfTheUSA` masyve.

```
  for ($counter=1; $counter<51; $counter++) {
  echo"<INPUT TYPE=HIDDEN NAME='HiddenState[]'
VALUE='$StatesOfTheUSA[$counter]'>";
  }
```

Po to jau galima sukurti Submit Query mygtuką ir uždaryti formą:

```
  echo "<INPUT TYPE=SUBMIT></FORM>";
```

Mūsų antrasis puslapis susideda iš dviejų duomenų šaltinių. Pirmieji gauti iš sąrašo laukelio ir turi vienos valstijos pavadinimą kurį pasirinko vartotojas. Ji saugoma kintamajame `$State`. Antri duomenys yra visas 50 valstijų sąrašas, alphabetine tvarka saugomas masyve `$HiddenState`. Tačiau, kadangi PHP darė šį masyvą jo indeksas prasideda 0 ir baigiasi 49. Tai nėra problema, paprasčiausiai reikia tai prisiminti. Mūsų pirmoji eilutė ir sprendžią šį klausimą. Dabar mes sukuriame masyvą su 50 valstijų sostinėmis. Tačiau norint, kad šis

masyvas sutaptu su indeksu esančiu `$HiddenState` masyve mums reikia, kad naujai kuriamas masyvas irgi prasidėtu nuo 0 ir taip iki 49:

```
$StateCapital = array (0 => "Montgomery", "Juneau",
"Phoenix", "Little Rock", "Sacramento","Denver","Hartford",
"Dover","Tallahasse", "Atlanta", "Honolulu",
"Boise", "Springfield","Indianapolis", "Des Moines",
"Topeka", "Frankfort", "Baton Rouge",
"Augusta","Annapolis","Boston", "Lansing", "Saint
Paul","Jackson", "Jefferson City", "Helena","Lincoln",
"Carson City","Concord", "Trenton","Santa
Fe", "Albany", "Raleigh","Bismarck","Columbus","Oklahoma
City", "Salem", "Harrisburg", "Providence",
"Columbia","Pierre", "Nashville", "Austin","Salt Lake
  City", "Montpelier","Richmond","Olympia","Charleston",
"Madison","Cheyenne");
```

Tai reiškia, kad masyvas `$HiddenState` ir masyvas `$StateCapital` sutampa savo indeksais. Visa kita programos dalis yra pakankamai paprasta. Mums tereikia vieno ciklo kuris vyktu nuo 0 iki 49.

```
for ($counter=0; $counter<50; $counter++)
{
```

Ciklo viduje mums reikia sąlygos elemento, kuris patikrintu ar dabartinė valstija sutampa su vartotojo pasirinktąją. Jei taip ir yra mes parodome `$StateCapital` reikšmę pasinaudodami indeksų tapatumu tarp masyvų:

```
if($HiddenState[$counter]==$State)
 {
  echo"The State capital is $StateCapital[$counter]";
 }
}
```

Tariant kitais žodžiais, jei `$HiddenState` turinys sutampa su vartotojo pasirinkta valstija, tada šis indeksas atitinka ir valstijos sostinės pavadinimą iš `$StateCapital` masyvo.

Mūsų programos patobulinimas

Tiesa pasakius, mes padarėme kiek gremėzdišką programą, nes norėjome pademonstruoti dviejų atskirų masyvų sąryšį per indeksą. Jei jums tikrai reiktu daryti tokio tipo programą, būtu daug paprasčiau padaryti asociacinį masyvą:

```
$StateCapital = array ("Alabama" => "Montgomery", "Alaska"
=> "Juneau", "Phoenix" =>"Arizona", "Arkansas" => "Little
Rock", "California" => "Sacramento","Colorado"
=> "Denver", "Connecticut" => "Hartford", "Delaware" =>
"Dover", "Florida" => "Tallahasse", "Georgia" => "Atlanta",
"Hawaii" => "Honolulu", "Idaho" => "Boise",
"Illinois" => "Springfield","Indiana" => "Indianapolis",
"Iowa" => "Des Moines", "Kansas" => "Topeka", "Kentucky" =>
"Frankfort", "Louisiana" => "Baton Rouge","Maine" =>
"Augusta","Maryland" =>"Annapolis","Massachusetts" =>
"Boston", "Michigan" => "Lansing", "Minnesota" => "Saint
Paul","Mississippi" => "Jackson", "Missouri" => "Jefferson
City", "Montana" => "Helena","Nebraska" => "Lincoln",
"Nevada" => "Carson City","New Hampshire" => "Concord",
"New Jersey" => "Trenton","New Mexico" => "Santa Fe", "New
York" => "Albany", "North Carolina" => "Raleigh","North
Dakota" => "Bismarck","Ohio" => "Columbus","Oklahoma" =>
"Oklahoma City", "Oregon" => "Salem", "Pennsylvania" =>
"Harrisburg", "Rhode Island" => "Providence", "South
Carolina" => "Columbia","South Dakota" =>"Pierre",
"Tennessee" => "Nashville", "Texas" => "Austin","Utah" =>
"Salt Lake City", Vermont" => "Montpelier","Virginia" =>
"Richmond","Washington" => "Olympia","West Virginia" =>
"Charleston", "Wisconsin" => "Madison","Wyoming"
=>"Cheyenne");
```

Tad jus galėtumėte pavaizduoti sostinę pasinaudodami valstijos pavadinimo kintamuoju `$State`, kaip indeksu.

Judėjimas per ne-nuoseklų masyvą

Taigi, kaip jūs matėte yra pakankamai lengva judėti per nuoseklų masyvą, kurio elementai buvo sudėti iš eilės (masyvo indeksas yra didėjantis nuosekliai). Bet kaip būti kai turime tokio tipo masyvą:

```
$Array[56993]="absolutely huge";
$Array[1]="quite small";
$Array[499]="quite big";
```

Ne-nuoseklūs masyvai nėra tokia didelė problema, kaip gali pasirodyti iš pražių, nes visi mūsų jau aptarti metodai veikia. Jei duomenys saugomi ne iš eilės, tai nėra svarbu PHP, nes jie yra suprantami kaip iš eilės einantys elementai. Vienintelė problema yra ta, kad tikrinant visus masyvo elementus iš eilės jūs galite rasti daugybe tuščių indeksų, kai tuo tarpu tik trys iš jų turi reikšmes, kaip kad pateiktame pavyzdyje.

Current ir Key funkcijos

PHP naudoja **žymeklį** (pointer) taip nustatydama kurioje masyvo vietoje ji yra, kai yra judame per masyvą. Šitas žymeklis nurodo į elementą kuris šiuo metu yra naudojamas skripto. Jūs galite sužinoti kuris masyvo elementas šiuo metu naudojamas pasinaudodami `current()` funkcija, ir jūs galite sužinoti šio elemento indeksą su `key()` funkcija.(Key yra kitas indekso pavadinimas.)

Mes pasinaudosime kodo dalimis parodydami kaip veikia `current() ir key()` funkcijos. Dabar įsivaizduokite kas atsitiktu jei jūs pridėtumėte dar vieną reikšmę tokiam masyvui, ir nurodytumėte, kad PHP tai padarytu automatiškai. Kokį indeksą pasirinktu PHP?

```
$Director[4]="Orson Welles";
$Director[1]="Carol Reed";
$Director[93]="Fritz Lang";
$Director[24]="Jacques Tourneur";
```

Tai mes galime sužinoti pridėja keletą kodo linijų. Jos parodo šiuo metu naudojama `$Director[],` indeksą:

```
$CurrentIndexValue = key($Director);
echo ($CurrentIndexValue);
```

`Key()` funkcija parodo reikšmę 4. Kodėl? Nes jis gražina indekso reikšmę pirmo elemento kurį mes patalpinome į masyvą – tai yra "Orson Welles" esantis 4 indekse. Jei naudosite `current()` funkcija ji gražins reikšmę "Orson Welles":

```
$CurrentContents = current($Director);
echo ($CurrentContents);
```

Dabar jei patalpinti sekančią eilute į masyvą:

```
$Director[]="Alfred Hitchcock";
```

"Alfred Hitchcock" masyve būtu patalpintas su indekso reikšme 94. Kaip mes galime patikrinti kokia yra naujojo elemento reikšmė, turint omenyje, kad `key() ir current()` funkcijos gražina informacija apie pirmąjį masyvo elementą?

Next() ir Prev() funkcijos

Norint sužinoti naujai įdėto elemento indeksą jums reikės naudotis `next()` ir `prev()` funkcijomis. Šios funkcijos jums leidžia judėti per masyvą, perkeldamos žymeklį į kitą ar į prieš tai buvusį masyvo elementą. Jiems abiems tereikia tik masyvo vardo kaip argumento. Dabar grįžkite prie jau anksčiau daryto masyvo:

```
$Director[4]="Orson Welles";
$Director[1]="Carol Reed";
$Director[93]="Fritz Lang";
$Director[24]="Jacques Tourneur";
$Director[]="Alfred Hitchcock";
```

Mes iškviečiame `next()` funkcija, prieš patikrindami esamą indeksą ir esamo elemento reikšmę:

```
next($Director);
$CurrentIndexValue = key($Director);
echo ($CurrentIndexValue);
```

Gauta reikšmė yra 1, ir `current()` funkcija gražins vardą "Carol Reed". Jei iškviesime `next()` dar tris kartus:

```
next($Director);
next($Director);
next($Director);
next($Director);
$CurrentIndexValue = key($Director);
echo ($CurrentIndexValue);
```

Reikšmė bus 94. Jei dabar iškviestumėme `current()` funkcija:

```
$CurrentContents = current($Director);
echo ($CurrentContents);
```

Mūsų naršyklė parodytu Alfred Hitchcock.

`Prev()` funkcija yra naudojama panašiu būdu. Mes pasinaudosime mūsų prieš tai buvusiu pavyzdžiu ir parašysime vieną `prev()` einanti po keturių `next()`:

```
next($Director);
next($Director);
next($Director);
next($Director);
prev($Director);
$CurrentIndexValue = key($Director);
echo ($CurrentIndexValue);
```

Šį kartą mes gausime 24, kuris yra indeksas su juo susijusios reikšmės "Jacques Tourneur". Taip yra todėl, kad mes pasistūmėjome į prieki per keturis elementus ir sugrįžome atgal per vieną. Tai gana tiesmukiškas pavyzdys, nors gali kilti klausimas, kas atsitiks jei mes pasistūmėsime į priekį su `next()` esančiu už paskutinio masyvo elemento arba `prev()` prieš pirma elementą?

```
prev($Director);
$CurrentIndexValue = key($Director);
echo ($CurrentIndexValue);
```

Atsakymas: nieko. Šis kodas negražina jokios reikšmės. Tai nesukels klaidos (kaip kad yra kai kuriuose kituose programavimo kalbose); žymeklis paprasčiausia pasislinks už masyvo ribos. Tačiau mes jau negalėsime grįžti atgal:

```
prev($Director);
next($Director);
next($Director);
```

```
$CurrentIndexValue = key($Director);
echo ($CurrentIndexValue);
```

Tai vis dar neduos jokio atsakymo.

Taigi mes matome, kad navigacija masyve nėra tokia jau sudėtinga. Nauja reikšmė yra patalpinama į pirmą neužimtą indekso poziciją po paskutinės didžiausios reikšmės. Taigi jei didžiausia reikšmė yra 34, kita indekso reikšmė būtu 35. Norint judėti masyvu mes naudojamės `next()` ir `prev()` funkcijomis. Mūsų dabartinės pozicijos parodymui naudojama `current()` ir `key()` funkcijos.

List ir Each funkcijos

Jei jūs judate per ne-nuoseklų masyvą, yra funkcijų kurios tikrai palengvins jums gyvenimą. Vietoj to, kad daryti ciklus su daugybę tuščių reikšmių jūs galite pasinaudoti `list()` ir `each()` funkcijomis, kurios gražins tik tuos masyvo elementus kurie turi reikšmes. Tai leidžia jums pavaizduoti visus masyvo elementus mažiausiais vargais. Jūs galite naudoti `while` ciklą norėdami atlikti tokią užduotį:

```
while (list(IndexValue,ElementContents) =
each(ArrayName))
```

Tai išsiverčia taip: kiekvienam masyvo `Arrayname` elementui nustatyk `IndexValue` lygią elemento indeksui ir `ElementContents` lygu elemento reikšmei. Jei jūs norite gauti tik indeksą ar tik reikšmę jūs galite rašyti tik reikiamus atributus:

```
while (list(IndexValue) = each(ArrayName))
```

arba:

```
while (list(,ElementContents) = each(ArrayName))
```

Mūsų autorių kodo pavyzdžiui mes galime parašyti tokį pavyzdį, kuris parašys visas masyvo reikšmes:

```
while (list($ElementIndexValue, $ElementContents) =
each($Director))
    {
     echo "<BR>$ElementIndexValue - $ElementContents";
    }
```

Jums nereikia rašyti būtent tokio pavadinimo kintamuosius `$ElementIndexValue` ir `$ElementContents`, tai daroma, kad kodas būtu lengviau skaitomas. Jūs laisvai galite naudoti tokias reikšmes:

```
while (list($MickeyMouse, $DonaldDuck) = each ($Director))
   {
    echo "<BR>$MickeyMouse - $DonaldDuck";
   }
```

Viskas ką jums reikia prisiminti yra tai, kad `list()` gražina visų pirma indekso po to elemento reikšmę. tai leidžia pasinaudoti supaprastintais programavimo elementais, kurie taip pat tinka ir tuo atveju jei masyvo indeksas nėra skaitinės reikšmės.

Judėjimas per tekstinio indekso masyvus

Taisyklės leidžiančios judėti per asociacinius masyvus yra tapačios kaip ir naudojamos skaitiniuose indeksuose, bet yra ir keletas skirtumų. Pirmasis

skirtumas yra tas, kad žemiau esantis kodas puikiai veikęs skaitiniame masyve dabar sukurs skaitinio indekso masyvą:

```
$StateCapital["ga"]="Atlanta";
$StateCapital["il"]="Springfield";
$StateCapital["ca"]="Sacramento";
$StateCapital[] = "Cheyenne";
```

Reikšmė "Cheyenne" bus išsaugota masyve kaip `$StateCapital[0]`. Tai nestebina, jei prisiminsite, kad PHP neturi jokio supratimo kokią indekso reikšmę jūs norite sukurti, kuri gali būti ir "WY", ir "AB", ar 4563, tad vietoj to PHP suras didžiausiai skaitinę reikšmę. kadangi mes dar nenaudojome jokių skaičių, didžiausia reikšmė bus 0.

`Current()` ir `key()` funkcijos vis tiek veiks taip kaip jūs ir numanote. Jei pabandysite sekantį kodą:

```
$WhatState = current($StateCapital);
$WhatAbbreviation = key($StateCapital);
echo "$WhatAbbreviation - $WhatState";
```

Jūs išsiaiškinsite, kad funkcijos gražina pirmojo į masyvą įvesto elemento reikšmę, kuris yra "ga" turintis "Atlanta" reikšmę.

Ir jūs galite judėti per masyvą kaip ir anksčiau.

```
$StateCapital["ga"]="Atlanta";
$StateCapital["il"]="Springfield";
$StateCapital["ca"]="Sacramento";
$StateCapital[] = "Cheyenne";
next($StateCapital);
$WhatState = current($StateCapital);
$WhatAbbreviation = key($StateCapital);
echo "$WhatAbbreviation - $WhatState";
```

Šiuo atvėju atsakymas būtu il - Springfield. Funkcijos `list()` ir `each()` taip pat veikia tokiu pačiu principu kaip ir masyvuose su skaitiniu indeksu. Mes galime parašyti sekantį kodą:

```
$StateCapital = array ("ga" => "Atlanta", "il" =>
"Springfield", "ca"=>"Sacramento", "wy" => "Cheyenne");
```

Tada mes vis dar galime pasinaudoti abiejomis funkcijomis ir pasinaudoja `list()` funkcija parodyti atsakymą kaip ir anksčiau:

```
  while (list($StateAbbreviation, $StateName) = each
($StateCapital))
    {
      echo "<BR>$StateAbbreviation - $StateName";
    }
```

Tai padarys abreviatūrų ir pavadinimų sąrašą, kurios bus pavaizduotos ta tvarka, kokia buvo sukurtos:

ga - Atlanta
il - Springfield
ca - Sacramento
wy - Cheyenne

Mes apžvelgėme šiuos du skirtingus masyvo tipus ir sužinojome kaip PHP judės per jas indekso eiliškumu, nepriklausomai nuo sukūrimo eiliškumo. Kas atsitiks jei jūs norėsite sukurti arba išsaugoti kitokį eiliškumą?

Masyvų rūšiavimas

Yra keletas funkcijų kurias pateikia PHP masyvų rūšiavimui. Mes apžvelgsime penkias dažniausiai naudojamas. Jos visos veikia sąveikaudamos su `list()` ir `each()` funkcijomis kurias mes ką tik patarėme.

sort()

`Sort` funkcija yra pati pagrindinė rūšiavimo funkcija. Ji paima masyvo reikšmes ir surūšiuoja jas pagal alfabetą. `Sort()` funkcijai tereikia masyvo pavadinimo kurį ji turi surūšiuoti:

```
sort(ArrayName)
```

Mes sukūrėme autorių masyva:

```
$Director = array ("Orson Welles","Carol Reed","Fritz
Lang","Jacques Tourneur");
```

Tada mes galime juos surūšiuoti parašę masyvo vardą į `sort()` funkcija:

```
sort($Director);
```

Norint pamatyti šios funkcijos efektą mes teturime vieną galimybę, vėl pasinaudoti `list()` ir `each()` funkcijomis.

Mes jau pareiškėme, kad seka kuria elementai yra saugomi masyve yra ta pati kaip seka kuria jie buvo sukurti. Su aukščiau esančiu masyvu, mes galime tikėtis tokios sekos:

```
$Director[0]= "Orson Welles"
$Director[1]= "Carol Reed"
$Director[2]= "Fritz Lang"
$Director[3]= "Jacques Tourneur"
```

Tačiau po `sort()` panaudojimo, mes gauname kitokią struktūrą:

```
$Director[0]= "Carol Reed"
$Director[1]= "Fritz Lang"
$Director[2]= "Jacques Tourneur"
$Director[3]= "Orson Welles"
```

Jūs galite tai patikrinti, pasinaudoja mūsų ankstesne kodo dalimi, kuri parodo masyvo reikšmes ekrane:

```
while (list($IndexValue, $DirectorName) = each
($Director))
    {
     echo "<BR>$IndexValue - $DirectorName";
    }
```

Taigi mūsų masyvas buvo surūšiuotas pagal alfabetą ir atitinkamai pakeisti indeksai. Tačiau kas atsitiks jei mes pabandysime surūšiuoti asociacinį masyvą?

asort()

`Asort()` funkcija paima masyvą sukurta su raidiniu indeksu ir surūšiuoja pagal reikšmes. Sugrįžkime prie mūsų jau nagrinėto pavyzdžio:

```
$StateCapital = array ("ga" => "Atlanta", "il" =>
"Springfield",
"ca"=>"Sacramento", "wy" => "Cheyenne");
```

Bus sukurtas tokio tipo masyvas:

```
$StateCapital["ga"]= "Atlanta"
$StateCapital["il"]= "Springfield"
$StateCapital["ca"]= "Sacramento"
$StateCapital["wy"]= "Cheyenne"
```

Jei atliksite rūšiavimą tokiu būdu:

```
sort($StateCapital);
```

Tada gausite tokį masyvą:

```
$StateCapital[0]= "Atlanta"
$StateCapital[1]= "Cheyenne"
$StateCapital[2]= "Sacramento"
$StateCapital[3]= "Springfield"
```

Tariant kitais žodžiais raidinės indekso reikšmės yra pakeičiamos skaitinėmis – nelabai patogu ar ne?

Tačiau jei jūs vietoj to pasinaudosite `asort()` funkcija:

```
asort($StateCapital);
```

Seka bus tokia:

```
$StateCapital["ga"]= "Atlanta"
$StateCapital["wy"]= "Cheyenne"
$StateCapital["il"]= "Sacramento"
$StateCapital["ca"]= "Springfield"
```

Šį kartą, elementai buvo išdėstyti alfabetiškai, tačiau mes išlaikėme ir susijusį indeksą. Vėl gi jūs galite pasinaudoti `list()` ir `each()` norėdami parodyti rezultatus:

```
while (list($StateAbbreviation, $StateName) = each
($StateCapital))
    {
      echo "<BR>$StateAbbreviation - $StateName";
    }
```

PHP taip pat pateikia įrankių ir atvirkštiniam rūšiavimui. Pažiūrėkime.

rsort() ir arsort()

Šios dvi funkcijos veikia panašiu būdu kaip ir `sort()` ar `asort()`. Vienintelis skirtumas yra tas, kad jos gražima rezultatus surūšiuotus atvirkštine alfabeto tvarka. Jūs galite pasinaudoti `rsort()` norėdami priešingai surūšiuoti mūsų autorių sąrašą:

```
$Director = array ("Orson Welles","Carol Reed","Fritz
Lang","Jacques Tourneur");
rsort($Director);
```

Jūs taip pat galite iškviesti `arsort()` priešingai surūšiuoti valstijų sostines:

```
$StateCapital = array ("ga" => "Atlanta", "il" =>
"Springfield",
"ca"=>"Sacramento", "wy" => "Cheyenne");
arsort($StateCapital);
```

Mes nežiūrėsime kokie yra atsakymai, iš to ką jūs jau žinote turite patys mokėti tai padaryti.

Yra dar vienas rūšiavimo tipas kurį mes turime apžvelgti, tai asociacinių masyvų rūšiavimas pagal indeksą

ksort()

`Ksort()` funkcija gali atlikti tokio tipo rūšiavimą. Ji atliekama tokiu pačiu būdu kaip ir kitos rūšiavimo funkcijos, bet vietoj to ji pagal abėcėlę surūšiuoja indekso reikšmes. Mūsų sostinių pavyzdžiu:

```
  $StateCapital = array ("ga" => "Atlanta", "il" =>
"Springfield",
  "ca"=>"Sacramento", "wy" => "Cheyenne");
  ksort($StateCapital);
```

`Ksort()` funkcija paradytu sekančius pakeitimus:

```
  $StateCapital["ca"]= "Sacramento"
  $StateCapital["ga"]= "Atlanta"
  $StateCapital["il"]= "Springfield"
  $StateCapital["wy"]= "Cheyenne"
```

Įvairios masyvo funkcijos

Yra didžiuli kiekis funkcijų kuriomis leidžia naudotis PHP, taip padarydama darbą labiau efektyvu. Jų yra taip daug, kad mes paprasčiausiai neturime laiko jas kiek plačiau apžvelgti, tačiau yra keletas kurias jūs tikrai turite žinoti, nes mes naudosime jas šioje knygoje ateityje.

array_push() ir array_pop()

Mes jau sakėme, kad jei jau turite masyvą ir į jį pridedate naują elementą nenurodydami indekso reikšmės, naujasis elementas gaus viena reikšme didesnį indeksą nei iki tol buvęs didžiausias indeksas. Tariant kitais žodžiais, jei mes pridėsime naują elementą į `$Director` masyvą žemiau:

```
  $Director[4]="Orson Welles";
  $Director[1]="Carol Reed";
  $Director[93]="Fritz Lang";
  $Director[24]="Jacques Tourneur";
  $Director[]="Alfred Hitchcock";
```

Naujasis elementas gaus indeksą 94. Funkcija `array_push()` daro tą patį dalyką. Norint įtraukti "Alfred Hitchcock" į masyvą mes rašytumėme:

```
  array_push($Director, "Alfred Hitchcock");
```

Tam naudojami du argumentai, pirmasis tai masyvo pavadinimas, o antras tai reikšmė kurią norite įtraukti į masyvą. Tačiau ši funkcija gali būti naudojama ir keletos elementų įtraukimui, juos išvardinant:

```
  array_push($Director, "Alfred Hitchcock", "FW Murnau",
"Akira Kurosowa");
```

`Array_pop()` funkcija daro atvirkščiai, pašalindama paskutinį įdėtą į masyvą elementą. Tad jei mes parašytumėme sekanti kodą:

```
  $Director[4]="Orson Welles";
  $Director[1]="Carol Reed";
  $Director[93]="Fritz Lang";
  $Director[24]="Jacques Tourneur";
  array_pop($Director);
```

Tai pašalintu elementą turinti indekso reikšmę 24, kurio reikšmė "Jacques Tourneur", paliekant tris elementus masyve. Atkreipkite dėmesį, kad skirtingai nei `array_push()`, `array_pop()` reikalingas tik vienas elementas: masyvo pavadinimas.

Abi funkcijos traktuoja masyvą kaip "**kūgį**": jūs patalpinate elementą at pačio viršaus, virš anksčiau ten esančių, taip pat jie ir pašalinami.

Implode ir Explode

Abi šios funkcijos nėra tokios griaunančios kaip jūs angliškos reikšmės. Jie leidžia išsaugoti viso masyvo reikšmes išsaugoti kaip eilute arba atvirkščiai: padalinti ilgą eilutę į gabalus ir šiuos gabalus patalpinti į masyvą kaip atskirus elementus.

`Implode()` funkcija paima jau esamą masyvą kaip argumentą, ir sujungia kiekvieną masyvo elementą į eilutę. Jūs taip pat galite nurodyti ir specialų atribojimo argumentą funkcijoje, kuris naudojamas atskirti kiekvieną masyvo elementą eilutėje:

```
$StringName = implode("delimiter", $ArrayName);
```

Jie paimsime mūsų sostinių pavyzdį, mes galime sujungti visas sostimes kartu, atskirdami jas brūkšneliu:

```
$StateCapital ["ca"]= "Sacramento";
$StateCapital ["ga"]= "Atlanta";
$StateCapital ["il"]= "Springfield";
$StateCapital ["wy"]= "Cheyenne";
$AllCapitalsSeparatedByADash = implode("-",
$StateCapital);
echo ($AllCapitalsSeparatedByADash);
```

`Echo` selementas gražins: Sacramento-Atlanta-Springfield-Cheyenne.

Atvirkštinė funkcija yra atliekama `explode()`. Sakykime mes turime eilutę kurioje mums reikalingi elementai yra atskirti bendru atribojimo ženklu, tokiu kaip brūkšnys ar tarpas. Mes galime suskaidyti eilutę į atskirus gabalus ir kiekvieną naują gabalą patalpinti į masyvą. Vėlgi, `explode()` paima du argumentus: eilutė (arba eilutės kintamasis), ir atribojimo argumentas:

```
$ArrayName = explode("delimiter", $StringName);
```

Mes atversime atgal mūsų prieš tai buvusį pavyzdį:

```
$AllCapitalsSeparatedByADash = "Sacramento-Atlanta-
Springfield-Cheyenne";
$Capitals = explode("-", $AllCapitalsSeparatedByADash);
```

Norint pažiūrėti koks masyvas buvo sukurtas, mes galime pasikliauti jau anksčiau šiame skyriuje išmoktu kodu:

```
$AllCapitalsSeparatedByADash = "Sacramento-Atlanta-
Springfield-Cheyenne";
$Capitals = explode("-", $AllCapitalsSeparatedByADash);
while (list($IndexValue, $CapitalName) = each ($Capitals))
 {
  echo "<BR>$IndexValue $CapitalName";
 }
```

Tai turėtu gražinti:

0 Sacramento
1 Atlanta
2 Springfield
3 Cheyenne

Taigi tai patvirtina, kad eilutė pavadinta `Capitals` buvo sukurta, ir indeksas buvo pradėtas nuo nulio. Kiekvienas eilutės elementas buvo paeiliui įtrauktas į masyvą. Kitaip tariant, `explode()` sukūrė masyvą kuris yra identiškas šiam PHP kodui:

```
Capitals[0]="Sacramento";
Capitals[1]="Atlanta";
Capitals[2]="Springfield";
Capitals[3]="Cheyenne";
```

Abi šios funkcijos yra labai naudingos, ir mes jas naudosime ateityje šioje knygoje.

HTTP_GET_VARS ir HTTP_POST_VARS

Paskutiniai du elementai apie kuriuos mes pašnekėsime yra ne funkcijos, o masyvai. Jie jau buvo sukurti PHP ir vadinasi `HTTP_GET_VARS` bei `HTTP_POST_VARS`. Kai mes šnekėjome apie formas 3 Skyriuje, mes paminėjome du pagrindinius metodus perduoti duomenis iš formos naudojantis `GET` ar `POST`. Mes pabrėžėme, kad PHP automatiškai sukuria kintamuosius už mus pagal formos elementų vardus. Kitaip tariant žemiau esančios formos laukelis taps kintamuoju `$TextBox32`:

```
<INPUT TYPE=TEXT NAME="TextBox32">
```

Šitas teiginys yra teisingas. Tačiau yra situacijų kai kintamieji kurie jūsų manymu turi būti sukurti automatiškai nėra sukuriami. Pavyzdžiui jei naudositės kažkieno teikiamom Interneto paslaugomis bei web serveriu. Norint kad šie kintamieji būtu kuriami automatiškai, konfigūracijos punktas `register_globals` turi būti nustatytas kaip „on“ jūsų `php.ini` faile. Tai daroma automatiškai instaliuojant PHP jūsų serveryje, tačiau kai kurie puslapiu talpinimo tiekėjai saugumo sumetimais nustato `register_globals` kaip „off“. Rezultatas tas, kad kintamasis susyjąs su viršuje esančiu formos laukeliu `$TextBox32` bus tuščias. Jie jūsų tiekėjas taip padarė neišsigąskite, kadangi informacija kuriuos jums reikia vis tiek pasiekiama per du masyvus pavadintus `HTTP_GET_VARS` ir `HTTP_POST_VARS`. Mes galime pasiekti informacija štai taip. Sakykime mes žinome kintamąjį kuris buvo pasiųstas naudojantis `GET` metodu:

```
<FORM METHOD=GET ACTION="transfer.php">
<INPUT TYPE=TEXT NAME="TextBox32">
```

PHP skripte puslapyje `transfer.php`, mes galime gauti teksto laukelio duomenis iš asociacinio masyvo `HTTP_GET_VARS`. Indekso vardas yra elemento pavadinimas "TextBox32" − vardas kurį mes sukūrėme su HTML elementais. Taigi jei norėsite gauti duomenis paprasčiausiai pasinaudokite tokiu kodu:

```
echo HTTP_GET_VARS["TextBox32"];
```

Tai veikia tiek su `GET` tiek su `POST`. Panašiai mes galime perduoti šias reikšmes ir kintamąjam:

```
$TextBox32 = HTTP_GET_VARS["TextBox32"];
```

Taigi tik viena papildoma kodo linija, ir mes apėjome problemą, kuri atsitinka kai `register_globals` yra išjungti.

Tačiau, tai gali būti tik problemų pradžia. Jei jūsų web serverio tiekėjas yra pakvaišęs dėl saugumo jis gal būt išjungė ir nuostata pavadinta `track_vars`, kuri atsakinga už šiuos du masyvus. Šiuo atveju parašykite savo tiekėjui laišką prašydami pakeiti konfigūraciją, arba pasirinkite kitą tiekėją.

Mes naudosimės ir šiais dviem masyvais šios knygos ateityje.

Daugiamačiai masyvai

Yra įmanoma sukurti masyvą masyve. Tokie „monstrai“ žinomi kaip daugiamačiai masyvai ,jie naudingi kai turimai informacijai reikia dvigubo indeksavimo, pavyzdžiui koordinatės žemėlapyje. Jūs galite kurti masyvus masyvuose tol kol PHP užteks operatyvios atminties, o tai tikriausiai yra daugiau nei 100 masyvų vienas kitame. (Jei galite sugalvoti situaciją kurioje jums reikės tokios struktūros, ir jei sugebėsite efektyviai ją pasinaudoti, jūs mažų mažiausiai esate PHP genijus)

Daugiamačiai masyvai yra daromi tokiu pat būdu kaip ir paprasti masyvai, išskyrus tai, kad jūs naudojate `array` struktūra kaip argumentą į patį save, štai taip:

```
ArrayName = array (index => array (Array contents))
```

Pavyzdys gali būti masyvas kuriame kiekvienas elementas reprezentuoja vieną žmogų, ir kiekvienas šis elementas savo ruožtu yra masyvas kuriame saugomi duomenys apie tą žmogų:

```
$PhoneDirectory = array ("John Doe" => array ("1 Long Firs
Drive","777-000-000"), "Jane Doe" => array ("4 8th and
East","777-111-111"));
```

Mūsų masyvas dabar turi elementus "John Doe" ir "Jane Doe"; ir kiekvienas elementas reprezentuoja masyvą su dviem elementas – adresu ir telefono numeriu. Norint šitai pavaizduoti ekrane jums reikės ir sudėtingesnio ciklo:

```
$PhoneDirectory = array ("John Doe" => array("1 Long Firs
Drive","777-000-000"), "Jane Doe" => array("4 8th and
East","777-111-111"));
  while (list($Person) = each($PhoneDirectory))
   {
     echo("<BR>$Person");
     while (list(,$PersonalDetails) = each
($PhoneDirectory[$Person]))
      {
        echo (" $PersonalDetails");
   }
  }
```

`<BR>` ir tarpas prieš `$PersonalDetails` yra `echo` elemento dalis, ir naudojama tik pagerinti vaizduojamos informacijos pateikimą. Daugiamačiai masyvai nėra labai dažnai naudojami, tad mes nesustosime ties šiuo klausimu.

Išbandykite – Masyvo galimybių sujungimas su praktiniu uždaviniu

1. Prikelkite savo teksto redaktorių ir parašykite:

```
<HTML>
<HEAD></HEAD>
<BODY>
<?php
echo"<FORM METHOD=POST ACTION='exam2.php'>";
$Student = array("Albert Einstein","Ivan The
Terrible","Napoleon","Simon
Bolivar","Isaac Newton");
while (list(,$Name)=each($Student))
 {
  echo "What grade did $Name get in Math?";
  echo"<BR><BR>"
```

```
  echo"<SELECT NAME='Math[]'>
  <OPTION>Grade A</OPTION>
  <OPTION>Grade B</OPTION>
  <OPTION>Grade C</OPTION>
  <OPTION>Grade D</OPTION>
  <OPTION>Grade E</OPTION>
  </SELECT>"
  echo"<BR><BR>";
  echo"<INPUT TYPE=HIDDEN NAME=Student[] VALUE='$Name'>";
 }
echo"<INPUT TYPE=SUBMIT></FORM>";
?>
</BODY>
</HTML>
```

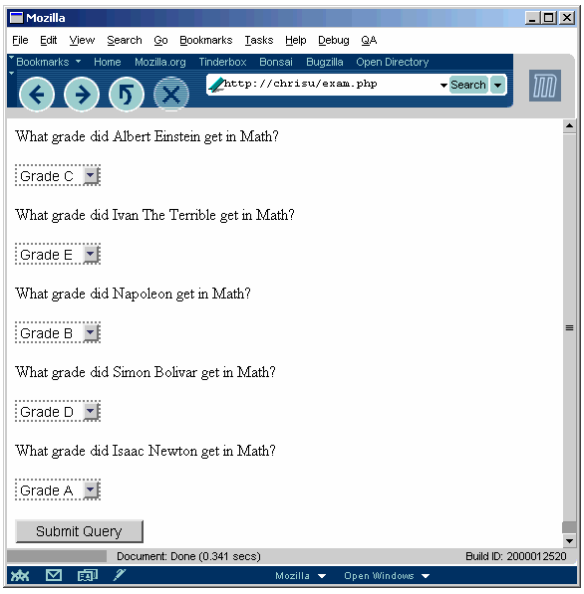
2. Išsaugokite kaip `exam.php`.

3. Uždarykite šitą failą ir sukurkite naują:

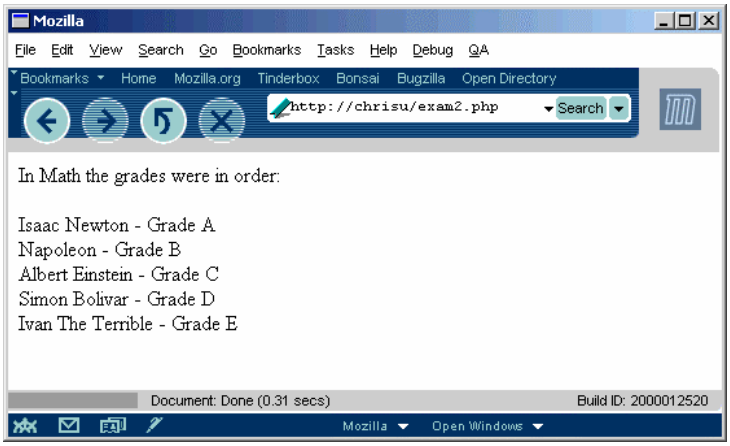
```
<HTML>
<HEAD></HEAD>
<BODY>
In Math the grades were in order:
<BR>
<?php
while (list($Index,$Value)=each($Math))
 {
  $GradeStudent[]=$Math[$Index].$Student[$Index];
 }
asort($GradeStudent);
while (list($Index,$Value)=each($GradeStudent))
 {
  echo "<BR>$Student[$Index] - $Math[$Index]";
 }
?>
</BODY>
</HTML>
```

4. Išsaugokite kaip `exam2.php`.

5. Atidarykite `exam.php` savo naršyklėje ir parašykite kokius nors pažymius:



6. Paspauskite ant Submit Query ir pažymiai bus surūšiuoti, ne tik pagal pažymius, bet ir pagal abėcėlę jei kurie nors mokiniai gavo tokius pačius pažymius:



Kaip tai veikia

Šitas pavyzdys nors ir yra gana trumpas, atlieka sudėtingus rūšiavimus, kaip jūs gal būt pastebėjote. Mes norėjome surūšiuoti pažymių masyvą, kuris indeksu atitinka studentų masyvą. Bet kaip mes galime susieti jau *surūšiuotą* pažymių masyvą su studentais? Čia reikia kiek pasukti galvą.

Pirmoji programa `exam.php` dinamiškai sukuria formą. Ji parūpina studentų vardus iš masyvo pavadinto `$Student`:

```
  echo"<FORM METHOD=POST ACTION='exam2.php'>";
  $Student = array("Albert Einstein","Ivan The
Terrible","Napoleon","Simon
  Bolivar","Isaac Newton");
```

Dabar mes pereiname prie `while` ciklo, kuris gauna kiekvieno studento varda ir pateikia klausimą:

```
  while (list(,$Name)=each($Student))
  {
```

Kiekvienam studentui sąraše, mes pateikiame klausimą klausianti koks jo pažymys:

```
  echo "What grade did $Name get in Math?";
```

dabar mes sukuriame sąrašo laukelį, kuris turi penkis galimus variantus, kiekvienas iš kurių atstovauja pažymiui, nuo A iki E. Norėdami išsaugoti pažymius mes sukuriame masyvą `$Math` kurį PHP gali perduoti sekančiam skriptui:

```
  echo"<SELECT NAME='Math[]'>
```

Mes parašome keletą linijų atskyrimo elementų, kad viskas atrodytu išvaizdžiau.

```
  echo"<BR><BR>";
```

Tačiau, kadangi mes sukūrėme masyvą pirmame puslapyje jis nebus prieinamas antrajame kuris gaus mūsų įvestus pažymius. Kaip apeiti šią problemą? Mes galime perduoti šiuos duomenis į antrąjį puslapį per paslėptą elementą. Šis elementas turi tokį patį pavadinimą kaip ir masyvas kuriame

saugomos studentų pavardės. Mes nurodome, kad elementas bus masyvas parašydami skliaustelius. Mes perduodame kiekvieno studento vardą ir tada uždarome ciklą:

```
echo"<INPUT TYPE=HIDDEN NAME=Student[] VALUE='$Name'>";
}
```

Pabaigoje mums reikia parašyti Submit mygtuką ir uždaryti formą:

```
echo"<INPUT TYPE=SUBMIT></FORM>";
```

Mūsų sekanti programa `exam2.php`, kuri gauna formos duomenis yra sudaryta iš ciklo, rūšiavimo ir dar viemo ciklo. Pirmasis ciklas yra skirtas susieti du masyvus gautus iš pirmojo puslapio. Mes tai padarome susijungdami studento vardą ir jo pažymį ir rūšiuojame rezultatus naujame masyve pavadintame `$GradeStudent`.

```
while (list($Index,$Value)=each($Math))
 {
  $GradeStudent[]=$Math[$Index].$Student[$Index];
 }
```

Mes žinome iš pirmosios programos, kad `Math` masyvo indeksas yra susyjąs su `$Student` masyvo indeksu. Gautas `$GradeStudent` masyvas atrodys štai taip:

```
Grade CAlbert Einstein
Grade EIvan The Terrible
Grade BNapoleon
Grade DSimon Bolivar
Grade AIsaac Newton
```

Dabar mes galime surūšiuoti `$GradeStudent` ir turime gauti tą tvarką kokios ir norime:

```
asort($GradeStudent);
```

Tačiau, vis tiek lieka problema kaip pavaizduoti atsakymus, nes dabar jie atrodo kiek keistokai. Mes galime pavaizduoti mūsų surūšiuotą pažymių sąrašą kartu su studentu vardais, naudodamiesi šia kodo dalimi:

```
while (list($Index,$Value)=each($GradeStudent))
 {
   echo "<BR>$Student[$Index] - $Math[$Index]";
 }
```

Ar suprantate ką mes padarėme? Atsiminkite, kad `$GradeStudent` yra surūšiuota pagal pažymius. Atsiminkite, kad rūšiuojant elementai buvo sukeisti vietomis, bet kiekvienas indeksas vis dar turi tokį patį turinį kaip ir prieš rūšiavimą. Taip pat mes jau matėme, kad kiekvienas elementas `$GradeStudent` masyve tėra paprasčiausias `$Math` ir `$Student` masyvo reikšmių sujungimas. Taigi, kas atsitiks jei mes pateiksime rezultatus iš `$Math` ir `$Student` masyvų, tiktai tai darysime pagal tvarką kurią nustato surūšiuotasis `$GradeStudent`? Kadangi visų trijų masyvų indeksai sutampa, mes išsiaiškinsime, kad `$Math` ir `$Student` masyvai yra vaizduojami pagal pažymius, taip kaip parodyta lentelėje:

$Index	**$Student**	**$Math Sorted**	**$GradeStudent**
4	`Isaac Newton`	Grade A	`"Grade AIsaac Newton"`
2	`Napoleon`	Grade B	`"Grade BNapoleon"`
0	`Albert Einstein`	Grade C	`"Grade CAlbert Einstein"`

3	Simon Bolivar	Grade D	"Grade DSimon Bolivar"
1	Ivan The Terrible	Grade E	"Grade EIvan The Terrible"

Mes žinome, kad naudojantis `list()` ir `each()` pateiks indekso reikšmes ta tvarka kokia kuria jie ir yra išdėstiti masyve. Taigi kodas pateiks elementus iš `$Math` ir `$Student` masyvų pagal pažymius. Tai užbaigia mūsų antrąją programos dalį.

Naujos ciklų ir masyvų galimybės PHP4

Atsiradus PHP4 pasirodė ir keletas naujų galimybių susijusių su ciklais ir masyvais. Šiame skyriuje mes apžvelgsime pagrindinius.

Masyvų multirūšiavimas (multisorting)

Pirmoji nauja galimybė kuria mes pristatysime yra `array_multisort()` funkcija, kuri pateikia kaip tą veikimą kurio mums reikėjo ankstesniame Matematikos pažymių pavyzdyje. Ji paima du masyvus kaip argumentus. Funkcija rūšiuoja pirmąjį masyvą ir pasižymi visus pasikartojančių reikšmių indeksus. Tada ji surūšiuoja šias pasikartojančias reikšmes pagal atitinkamų indeksų reikšmes antrajame masyve. Galiausiai funkcija rūšiuoja antrąjį masyvą pagal tą patį principą kaip surūšiavo pirmąjį. Paimkime mūsų prieš tai buvusį pavyzdį ir pažiūrėkime kaip jį galima modifikuoti. Mes galime pakeisti mūsų kodą `exam2.php` programoje, kad jis naudotu `$array_multisort()`:

```
<?php
array_multisort($Math,$Student);
while (list($Index,$Value)=each($Student))
  {
   echo "<BR>$Student[$Index] - $Math[$Index]";
  }
?>
```

Dabar išsiaiškinkime kaip vyksta visas procesas:

```
Albert Einstein - Grade A
Ivan The Terrible – Grade E
Napoleon – Grade D
Simon Bolivar – Grade D
Isaac Newton – Grade A
```

`$array_multisort()` funkcija pirma surūšiuotu `$Math[]` masyvą, parašydama A, A, D, D, E. kadangi mes turime dvi tokias pačias A ir D reikšmes, funkcija pasižymi A reikšmių indeksus (0 ir 4) ir kreipiasi į antrąjį masyvą `$Student[]` ir surūšiuoja šiuos du elementus 0 (Albert Einstein) ir 4 (Isaac Newton) pagal abėcėlę. Tas pats vyksta ir su D reikšmėmis. Kai funkcija baigia darbą su pirmuoju masyvu ji pereina prie antrojo ir atlieka tokius pačius veiksmus. Dabar abu masyvai visų pirma surūšiuoti pagal pažymius ir antra pagal pavardes.

Tačiau, tai nauja galimybė ir buvo pranešimų su jos naudojimu, todėl mes ir nenaudojome jos savo pavyzdyje.

foreach ciklas

Taip pat yra `for` ciklo plėtinys atsiradęs su PHP4. Tai paskutinis ciklo tipas kurį mes aptarsime: `foreach` ciklas. Jis naudojamas kai turime ciklą su nežinomu elementų skaičiumi. `Foreach` ciklas tesiasi iki masyvo pabaigos. Jis turi du formatus. Pirmasis yra toks:

```
foreach ($ArrayName As $ArrayItem)
 {
   execute the contents of these braces
 }
```

Tai reiškia, kad dėl kiekvieno elemento masyve bus atliekamas ciklas.

Antrasis formatas yra

```
foreach ($ArrayName As $ArrayIndexValue =>
$ArrayItem)
  {
   execute the contents of these braces
  }
```

Tai tas pats kaip ir pirmasis formatas tik tiek, kad čia mes galime operuoti ir masyvo elemento indeksu. Padarykime nedidelį `foreach` pavyzdį. Mes vėl pasinaudosime valstijų sąrašu ir paprasčiausiai pavaizduosime valstijų sąrašą kartu su indeksu kokį suteikė PHP.

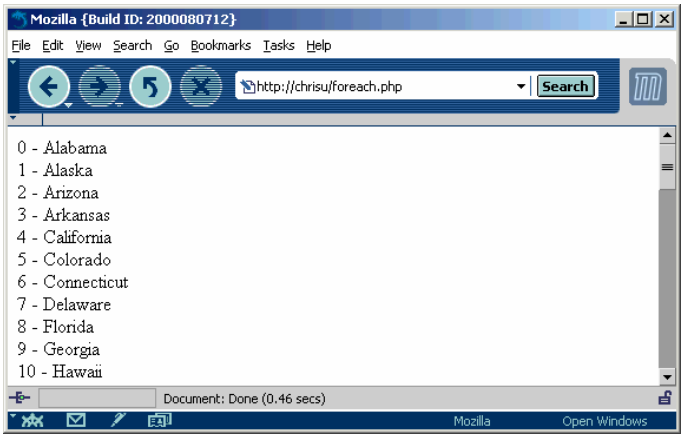
Išbandykite – foreach naudojimas

1. Su savo teksto redaktoriumi parašykite sekanti kodą:

```
<HTML>
<HEAD></HEAD>
<BODY>
<?
$StatesOfTheUSA = array ("Alabama", "Alaska", "Arizona",
"Arkansas", "California", "Colorado", "Connecticut",
"Delaware", "Florida", "Georgia", "Hawaii", "Idaho",
"Illinois", "Indiana", "Iowa", "Kansas", "Kentucky",
"Louisiana", "Maine", "Maryland", "Massachusetts",
"Michigan", "Minnesota", "Mississippi", "Missouri",
"Montana", "Nebraska", "Nevada", "New Hampshire", "New
Jersey", "New Mexico", "New York", "North Carolina", "North
Dakota", "Ohio", "Oklahoma", "Oregon", "Pennsylvania",
"Rhode Island", "South Carolina", "South Dakota",
"Tennessee", "Texas", "Utah", "Vermont", "Virginia",
"Washington", "West Virginia", "Wisconsin", "Wyoming");
foreach($StatesOfTheUSA As $StateIndex => $State)
 {
  echo "<BR>$StateIndex - $State";
 }
?>
</BODY>
</HTML>
```

2. Išsaugokite kaip `foreach.php`.

3. Atidarykite `foreach.php` savo naršyklėje:



4. Pažiūrėkite ar tikrai yra 50 valstijų, kurių reikšmės yra nuo 0 iki 49.

Kaip tai veikia

Tai negali būti paprasčiau. Mes visų pirma sukuriame masyvą visų 50 valstijų, kuris vadinasi `$StatesOfTheUSA`:

```
$StatesOfTheUSA = array ("Alabama", "Alaska", "Arizona",
"Arkansas", "California", "Colorado", "Connecticut",
"Delaware", "Florida", "Georgia", "Hawaii", "Idaho",
"Illinois", "Indiana", "Iowa", "Kansas", "Kentucky",
"Louisiana", "Maine", "Maryland", "Massachusetts",
"Michigan", "Minnesota", "Mississippi", "Missouri",
"Montana", "Nebraska", "Nevada", "New Hampshire", "New
Jersey", "New Mexico", "New York", "North Carolina", "North
Dakota", "Ohio", "Oklahoma", "Oregon", "Pennsylvania",
"Rhode Island", "South Carolina", "South Dakota",
"Tennessee", "Texas", "Utah", "Vermont", "Virginia",
"Washington", "West Virginia", "Wisconsin", "Wyoming");
```

Tada mes pateikiame masyvo vardą kaip pirmąjį argumentą `foreach` ciklui:

```
foreach ($StatesOfTheUSA As $StateIndex => $State)
```

Antrasis ir trečiasis argumentai yra kintamųjų vardai kuriuos mes patys sukūrėme, kurie laiko indekso ir atitinkamo elemento reikšmes. Mes parašome kiekvieną elementą vykdydami kodą:

```
{
echo "<BR>$StateIndex - $State";
}
```

Reikia pažymėti keletą aspektų. Pirmasis tas, kad masyvo elementas ir indeksas yra du atskiri kintamieji . pastebėkite, kad kiekvieną kartą vykdant ciklą vis kitos reikšmės yra priskiriamos kintamiesiems.

Antrasis dalykas yra tas, kad mes niekur programoje nenustatėme kiek elementų yra masyve. PHP pati tai nustatė, tai ir yra pagrindinis `foreach` privalumas. Tai leidžia jums judėti per masyvą kuriame gali nebūti jokiu skaitytinių ar raidinių elementų, taip pat gali būti praleisti masyvo elementai, bet jis neturi tikrinti kiekvieno masyvo indekso tik tuos kurie turi reikšmes. Mes galėtumėme pridėti prie sekančio kodo tokią eilutę:

```
$StatesOfTheUSA[100]="Atlantis";
```

Tačiau `foreach` ciklas neitu per 50-99 elementus, jis nagrinėja tik tuos kurie turi reikšmes.

Apibendrinimas

Šitas skyrius pristatė dvi skirtingas, bet susijusias temas: ciklus ir masyvus. Būdami konceptualiai skirtingi, ciklai ir masyvai yra tamprai susyja PHP kalboje, kaip kad mes ir parodėme. Jums reikia ciklu norint atlikti pasikartojančius veiksmus, ir viena iš tokių operacijų yra užpildyti dideles indeksuotu kintamųjų (masyvų) mases.

Pirmoje skyriaus dalyje mes ištyrėme tris ciklų tipus `while`, `do while` ir `for`. Mes nusprendėme, kad kiekvienas iš jų tinka skirtingoms užduotims, tad pasirinkimas kokį pasirinkti priklauso nuo situacijos. Tada mes pristatėme masyvus, ir pademonstravome metodus kaip su jais elgtis. Mes apžvelgėme kaip judėti per masyvus, net jei šie nėra skaitiniai ar ne-nuoseklūs. Mes taip pat išmokome rūšiuoti masyvus. Galiausiai mes trumpai apžvelgėme daugybinių masyvų koncepcija ir paragavome naujų ciklų ir masyvų galimybių kurias mums pristatė PHP4.

Sekančiame skyriuje mes tesime mūsų kelione po PHP fundamentalius dalykus, pristatydami struktūrinio programavimo kertinį akmenį – funkcijas.

6

Jūsų kodo organizavimas

Ankstesniuose savo skyriuose mes jau paminėjome, kad kodo pakartojimo panaudojimas yra geras dalykas, plačiau nepaaiškindami kodėl. Dar kartą, prieš pradėdami aptarinėti konkrečius PHP programavimo aspektus, mes turime pasinerti į fundamentalias programavimo struktūras.

Kai pirmą kartą pristatėme šakojimąsi, mes pacitavome vieną iš pagrindinių privalumų, kad mums nebereikia vykdyti kodo paeiliui. Tai tiesa, tačiau jūsų kodui tampant vis sudėtingesniam, jūs greitai suprasite, kad šakojimai ir ciklai yra nepakankami norint parašyti efektyvų kodą. Po ne didelio nukreipimo į sąlygos struktūras ar ciklus, jūs vis tiek grįšite prie pirmos toliau sekančios eilutės. Tiktai su **funkcijų** pristatymu, jūs galėsite rašyti tikrai mažas, nepriklausomas kodo dalis kurios gali būti iškviestos, iš bet kurios kodo vietos.

Funkcijos leidžia suglausti kodo dalį, lyg jis būtu atskira programa. Jūs galite jiems perduoti duomenis ir gauti atsakymus. Jūs galite iškviesti funkcijas bet kada jūsų kodo veikimo metu, taip nukreipdami kodo vykdymą kita linkme. Mes taip pat apžvelgsime kaip funkcijos gali veikti viena kitoje, vėl ir vėl. Ir kaip mes galime įterpti atskirą tekstą ar PHP skriptą į jūsų web puslapį.

Šiame skyriuje aptarsime šias temas pagal eilę

- ❑ Pakartotinio kodo naudojimo privalumai
- ❑ Funkcijos
- ❑ Kaip iškviesti funkcijas web puslapyje
- ❑ Duomenų perdavimas į ir iš funkcijos
- ❑ The scope of variables inside and outside functions
- ❑ Nesting functions
- ❑ Include failai

Pakartotinio kodo naudojimo privalumai

Mes nesiruošiame jums čia pateikti plačios programinės įrangos rašymo paskaitos, bet reikia pažymėti, kad galimybė naudotis kodo dalimis daugiau nei kartą suteikia daug privalumų. Pirmas ir didžiausias privalumas, kad reikia rašyti mažiau kodo, o tai reiškia, kad mažiau jo reikia ir tikrinti. Tačiau tai ne visi privalumai. Kodo pakartotinis naudojimas leidžia jums patogiau struktūrizuoti jūsų programą.

Kompiuterių jaunystėje (devinto dešimtmečio pradžioje), dauguma programinių kalbų, neleido rašyti struktūrizuotų programų, ir tikrai neturėjo tokių galimybių kaip funkcijos. Tačiau jos leido nukreipti programos vyksmą pagal užgaida. Tipiškai, kiekviena programos kodo linija turėjo savo numerį, dažniausiai didėjantį 10, jei jūs ką užmiršite ir vėliau norėsite ką nors parašyti. Taip pat buvo liūdnai pagarsėjusi `GOTO komandą,` kuri nurodydavo peršokti į tam tikrą eilutę. Daugumos žmonių pirmoji programa atrodydavo štai taip:

```
10 PRINT" HELLO WORLD"
20 GOTO 10
```

Tai neišvengiamai sukeldavo amžiną ciklą spausdinanti HELLO WORLD ekrane, kol vartotojas nenutraukdavo šios nesamonės. Dideliose programuose, net keletas `GOTO` komandų padarydavo jas labai sunkiai suprantamas, nes kodo veikimo eiliškumo kaip ir nebuvo. Jos netgi vadintos **spageti kodu**. Mes dabar čia šnekėdami jaučiame savo pranašuma, tačiau ir kai kuriam mūsų sukurtam kodui trūksta aiškumo ir nuoseklumo – jokio skirtumo nuo spageti kodo. Kaip to išvengti? Atsakymas yra, tačiaus pasukime truputėli į šoną.

Modulizacija (modularization)

Kai jūs rašote programą, jūs visų pirma apmąstote užduotį kurią reikia atlikti, ir padalinate ją į keletą mažesnių užduotėlių. Galvokite apie šias užduotis kaip apie modulius. Dabartiniai programuotojai rašo kodą moduliais ir kompiliuoją šiuos modulius į vieną didelę programą. Ją lengviau testuoti, jei konkreti programos dalis daro tik vieną užuoti. Yra tik keletas reikšmių kurios pateikiamos moduliams ir gaunami keli atsakymai. Sakykime mes žinome, kad kažkoks modulis veikia be klaidų, ir tada duomenis perduodami kažkokiam kitam moduliui kuris pateikia klaidingus atsakymus, iš karto aišku kurioje kodo dalyje yra klaida ir galima žiurėti kur konkrečiai ji atsitinka.

Taip pat jei modulis atlieka konkrečią funkciją, jūs galite iškviesti tą moduli tiek kartų keik jums reikia atlikti tai užduočiai. Jūs galite galvoti apie modulius kaip apie mažas juodas dėžes, į kurias jūs įdedate duomenis ir išimate atgal, ir tol kol duomenys yra teisingi jums visai nerūpi kas vyksta viduje. PHP (ir daugelyje kitų programavimo kalbų) tokie „moduliai“ vadinami **funkcijomis**.

Functions

Kaip mes jau sakėme, funkcijos yra kodo dalys skirtos atlikti konkrečiom užduotim. Mes jau matėme, kad funkcijos gali paimti reikšmes, vadinamas **argumentais**, atlikti kažkokias operacijas ir tada gražinti kitą reikšmę. Funkcija perdaro bet kokius argumentus į naujus kintamuosius, vadinamus **parametrais**, kurie vėliau gali būti naudojami už funkcijos ribų. Mes jau plačiai naudojomės funkcijomis šioje knygoje, pavyzdžiui rūšiavimo operacijose, kur būdavo pateikiamas masyvas ir gaunamas atsakymas (jau surūšiuotas). Šiame skyruje mes išmoksime patys pasidaryti funkcijas. Mūsų funkcijos veiks tokiu pat principu kaip ir standartinės, tad kai mes parūpinsime joms argumentus, jos atliks veikspus ir suteiks mums atsakymus.

Funkcijų apibrėžimas ir iškvietimas

Norint apibrėžti funkciją reikia suteikti jis vardą. Norint tai padaryti naudojamas raktažodis `function`, ir po jo sekantis funkcijos vardas. Visi parametrai yra nurodomi skliaustuose po funkcijos vardo. Kodas kuris suformuoja kodo pagrindą yra patalpinamas riestiniuose skliaustuose. Abstrakčiai viskas veikia štai taip:

```
function funkcijosvardas (parametrai)
 {
  funkcijos kodas...
 }
```

Pažiurėkime į pavyzdį kaip PHP apskaičiuoja mokesčius funkcijoje:

```
function mokesciai ($Alga)
 {
  $Alga = $Alga - (($Alga/100)*20);
  return $Alga;
 }
```

Pati funkcija gali turėti tiek kodo eilučių kiek tik norite, bet paskutinėje eilutėje (norėdami gauti atsakymą) jūs turite patalpinti galutinę išraišką ar apskaičiavimą po `return` raktažodžio. Tai pasako funkcijai, kad jūs baigėte savo skaičiavimus ir funkcija čia pasibaigia. Taip pat tai gali reikšti, kad apskaičiuota reikšmė turi būti gražinta vėl į funkciją. Jums nebutinau reikia gražinti atsakymą, jūs galite įvykdyti kodą, kaip kad parodyta šitam HTML pavyzdyje:

```
function html_header($page)
 {
   print "\n<HTML>\n<HEAD>\n<TITLE>My Website ::: " .
$page .   "</TITLE>\n</HEAD>";
  print "\n<BODY>";
  return;
 }
```

`Return` raktažodis čia naudojamas parodyti funkcijos pabaiga, o ne tam kad būtu gauti atsakymai. Tačiau jei jūs norite parodyti funkcijos sugražintą rekšmę, jūs galite pasinaudoti `echo` komandą. `Return` pati nepavaizduoja atsakymo; ji tik persiunčia atsakymus (jei tokie yra). PHP kalboje funkcijos turinio vygdymas vadimamas funkcijos **iškvietimu**. Jūs galite funkcijai tiesiog pateikti skiačių:

```
echo (mokesciai(1600));
```

arba galite jai duoti jau sukurto kintamojo vardą:

```
$Alga=1600;
echo (mokesciai($Alga));
```

Kad būtu aiškiau, jūs gal būt norėsite priskirti funkcijos rezultatus kintamajam, o tada parodyti juos ekrane:

```
$Alga=1600;
$ApskaisiuotaAlga = mokesciai($Alga);
echo ($ApskaisiuotaAlga);
```

Jei norite funkcijai pateikti daugiau nei vieną argumentą, jūs turite atskirti kintamuosius kableliais:

```
function mokesciai ($Alga,$Procentai)
 {
  $Alga = $Alga - (($Alga/100)*$Procentai);
  return $Alga;
```

```
 }
```

Norint iškviesti funkcija, reikia suteikti jai du argumentus:

```
echo (mokesciai(1600,25));
```

arba du kintamuosius:

```
$Alga=1600;
$Procentai=25;
echo (mokesciai($Alga,$Procentai));
```

arba kaintamųjų ir reikšmių mišinys:

```
$Procentai=25;
echo (mokesciai(1600,$Procentai));
```

Jūs galite ir nesutekti jokiu argumentų funkcijai:

```
function mokesciai ()
 {
  $Alga = 2500;
  $Procentai = 15;
  $Alga = $Alga - (($Alga/100)*$Procentai);
  return $Alga;
 }
```

Gal jūs nežinote kur parašyti kintamuosius kurie bus naudojami kaip argumentai funkcijoje? Jie gali būti rašomi prieš funkcija:

```
<?php
$Alga=1600;  //Cia prasideda PHP kodo vykdymas
$Procentai=25;
echo (mokesciai($Alga,$Procentai));  //Cia iskvieciama
funkcija
function mokesciai($Alga,$Procentai)  //Cia funkcija
apibreziama
  {
   $Alga = $Alga - (($Alga /100)*$Procentai);
   return $Alga;
  }
?>
```

Bet taip pat lengvai jie gali eiti ir po funkcijos. Jei pažiūrėsite į visą pavyzdinę programą:

```
<?php
function mokesciai($Alga,$Procentai)  //Cia funkcija
apibreziama
  {
   $Alga = $Alga - (($Alga /100)*$Procentai);
   return $Alga;
  }
$Alga=1600;  //Cia prasideda PHP kodo vykdymas
$Procentai=25;
echo (mokesciai($Alga,$Procentai));  //Cia iskvieciama
funkcija
?>
```

PHP kodo vykdymas prasidės per programos vidurį, su pirmąją linija kuri nėra funkcijoje. Mūsų funkcija nenaudojama tol kol ji neiškviečiama paskutinėje eilutėje. Jei pašalintumėme paskutinę eilutę funkcija iš viso nebūtu vykdoma. Jei funkcijos nereikia jos galima iš viso neiškviesti. Arba naudotis ją tiek kartu kiek tik reikia, ką mes tuoj ir parodysime:

```
$Alga=1600;
$Procentai=25;
echo (mokesciai(2000,$Procentai));
echo (mokesciai($Alga,30));
echo (mokesciai(2000,30));
echo (mokesciai($Alga,$Procentai));
```

Galiausiai, jūs neapriboti naudoti tą pati kintamąjį už funkcijos ribų kaip ir argumentai. Mes darėme pavyzdžius su tokiais pačiais kintamaisiais, nes norėjome, kad viskas būtu kiek galima aiškiau:

```
$Karve=1600;
$Pienas=25;
echo (mokesciai($Karve,$Pienas));
function mokesciai($Alga,$Procentai)
 {
  $Alga = $Alga - (($Alga/100)*$Procentai);
  return $Alga;
 }
```

Svarbiausiai, kad funkcija paima argumentu kintamuosius ir priskiria jų reikšmes kintamiesiems kurie bus naudojami funkcijos viduje. Taigi `$Alga` įgauna `$Karve` reikšmę funkcijos viduje, o `$Proventai` įgauna `$Pienas` reikšmę.

Prieš darant pavyzdį, trumpai apibendrinkime.

- Funkcija turi turėti vardą
- Funkcijos paima argumentus kurie yra reikšmės ar kintamieji kurie rašomi skliausteliuose po funkcijos vardo
- Jei yra keletas parametrų, jie atskiriami kableliais
- Funkcijos kodas pateikiamas riestiniuose skliaustuose po funkcijos vardo ir parametrų
- Jūs turite naudoti `return` raktažodį funkcijos viduje, norėdami gauti rezultatą kurį būtu galima naudoti už funkcijos ribų
- Jie nėra rezultato kurį reiktu gražinti, `return` raktažodis paprasčiausiai nurodo funkcijos pabaiga
- Funkcijos nevykdomos, kol kur nors PHP kode nėra iškviečiamos
- Jūs galite iškviesti funkcija jai esant prieš ar po iškvietimo, tai reiškia, kad funkcija gali būti bet kurioje kodo vietoje
- Jūs galite iškviesti funkcija tiek kartų kiek tik norite

Dabar mes išnagrinėsime atostogų pavyzdį kurį jau darėme ankstesniame skyriuje ir panaudosime funkciją apskaičiuodami atostogų išlaidas. Kaip patys pamatysite mes parašysime daug mažiau kodo eilučių.

Išbandykite – paprastos funkcijos naudojimas

1. Atidarykite savo redaktorių ir susiraskite jau padaryta `holiday.html`, pakeiskite paryškintą liniją. Jei nebeturite `holiday.html` parašykite visą žemiau esantį kodą:

```
<HTML>
<HEAD></HEAD>
<BODY>
<B>Namllu Holiday Booking Form</B>
<FORM METHOD=GET ACTION="holiday3.php">
Where do you want to go on holiday?
<BR>
<BR>
<INPUT NAME="Destination" TYPE="Radio" VALUE="Prague">
Prague
<BR>
<INPUT NAME="Destination" TYPE="Radio" VALUE="Barcelona">
Barcelona
<BR>
<INPUT NAME="Destination" TYPE="Radio" VALUE="Vienna">
Vienna
<BR>
<BR>
```

```
What grade of hotel do you want to stay at?
<BR>
<BR>
<INPUT NAME="Grade" TYPE="Radio" VALUE="Three">
Three Star
<BR>
<INPUT NAME="Grade" TYPE="Radio" VALUE="Four">
Four Star
<BR>
<BR>
<INPUT TYPE=SUBMIT>
</FORM>
</BODY>
</HTML>
```

2. Išsaugokite kaip `holiday.html`.

3. Uždarykite šitą failą ir padarykite naują:

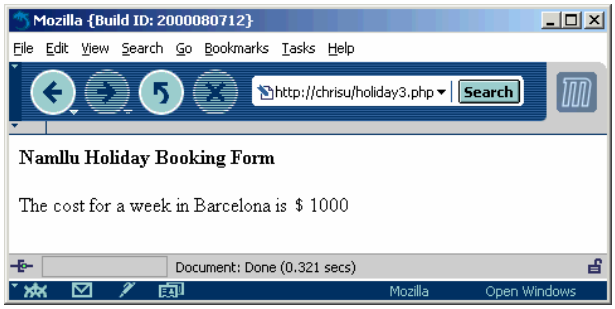
```
<HTML>
<HEAD></HEAD>
<BODY>
<B>Namllu Holiday Booking Form</B>
<BR>
<BR>
<?php
function Calculator($Price, $CityModifier, $StarModifier)
{
return $Price = $Price * $CityModifier * $StarModifier;
}
$Price=500;
$StarModifier=1;
$CityModifier=1;
$DestGrade = $Destination.$Grade;
switch($DestGrade) {
case "BarcelonaThree":
$CityModifier=2;
break;
case "BarcelonaFour":
$CityModifier=2;
$StarModifier=2;
break;
case "ViennaThree":
$CityModifier=3.5;
break;
case "ViennaFour":
$CityModifier=3.5;
$StarModifier=2;
break;
case "PragueThree":
break;
case "PragueFour":
$StarModifier=2;
break;
default:
$CityModifier=0;
echo ("Go back and do it again");
}
if ($CityModifier<>0)
{
echo "The cost for a week in $Destination is " . "$" .
Calculator($Price,$CityModifier,$StarModifier);
}
?>
</BODY>
</HTML>
```

4. Išsaugokite kaip `holiday3.php`.

5. Atidarykite `holiday.html` savo naršyklėje.

6. Pasirinkite keletą opcijų ir paspauskite Submit.

7. Jūsų pavyzdys turėtu veikti normaliai:



Kaip tai veikia

Atostogų forma turėtu būti labai pažystama, ji perduoda tik du kintamuosius: kelionės vietą (`$Destination`) ir viešbučio tipą (`$Grade`). Mes pradedame PHP skriptą nuo funkcijos kuri apskaičiuoja atostogų kainą:

```
function Calculator($Price, $CityModifier, $StarModifier)
 {
  return $Price = $Price * $CityModifier * $StarModifier;
 }
```

Ji paima tris parametrus – pagrindinę atostogų kainą, kainos skirtumą priklausomai nuo miesto, ir kainos skirtumą priklausomai nuo viešbučio lygio. Funkcija padaugina šias tris reikšmes ir gražina vieną reikšmę – bendrą atostogų kainą. Šie skaičiavimai ankstesniame pavyzdyje buvo atlikinėjami daug kartų, tuo tarpu čia mums tereikia išsikviesti funkcija ir pateikti jai duomenis. Visa kita programos dalis irgi neturėtu sukelti jums sunkumų. Mes sujungiame kintamuosius `$Destination` ir `$Grade`:

```
$DestGrade = $Destination.$Grade;
```

Priklausomai nuo jų reikšmės mes pasirenkame ir įvykdome tam tikrą kodo dalį. Kodas su `case` pasirinkimu paprasčiausiai priskiria reikšmes kintamiesiems `$CityModifier` ir `$StarModifier`:

```
switch($DestGrade) {
  case "BarcelonaThree":
  $CityModifier=2;
  break;
case "BarcelonaFour":
  $CityModifier=2;
  $StarModifier=2;
  break;
case "ViennaThree":
  $CityModifier=3.5;
  break;
case "ViennaFour":
  $CityModifier=3.5;
  $StarModifier=2;
  break;
case "PragueThree":
  break;
case "PragueFour":
  $StarModifier=2;
  break;
default:
  $CityModifier=0;
  echo ("Go back and do it again");
}
```

Mes padarėme viena ne didelį pakeitimą apačioje `default` opcijoje, pakeisdami kintamojo `$CityModifier` reikšmę į nulį, jei netiko nei viena iš sąlygų. Prieš vykdydami funkcija mes patikriname ar `$CityModifier` buvo nurodytas kaip nulis ar ne, jei nebuvo mes galime iškviest funkcija su `echo()` komandą, pasinaudodami sujungimo operatoriumi (tašku) pateikdami atsakymus:

```
if ($CityModifier<>0)
 {
  echo "The cost for a week in $Destination is " . "$" .
  Calculator($Price,$CityModifier,$StarModifier);
 }
```

Ši funkcija paima visus tris kintamuosius kaip argumentus ir pateikia atsakymą.

Funkcijos ir switch()

Jei žengsite dar vieną žingsnį toliau, nei padarėme mūsų paskutiniame pavyzdyje kur naudojome `switch()` elementą, labai naudinga pasirinkti funkcijas konkrečiam veiksmui su `switch()` elementu. Įsivaizduokime programą, kurioje priklausomai nuo mūsų pasirinkto maršruto tikslo paskaičiuojama reikalinga pinigų suma ir ji savo ruožtu paverčiama į vietinę valiutą:

```
switch ($City)
 {
  case "Oslo":
    echo(Norwegian_converter($MoneyAmount));
  break;
  case "Stockholm":
    echo(Swedish_converter($MoneyAmount));
  break;
  case "Copenhagen":
    echo(Danish_converter($MoneyAmount));
  break;
  default:
   echo $MoneyAmount;
    }
      function Norwegian_converter($MoneyAmount)
    {
      ...apskaičiuoja ir pateikia sumą Norvegiška valiuta
    }
     function Swedish_converter($MoneyAmount)
   {
     ...apskaičiuoja ir pateikia sumą Švediška valiuta
   }
     function Danish_converter($MoneyAmount)
   {
     ...apskaičiuoja ir pateikia sumą Daniška valiuta
   }
```

Tai tikrai didelis kodo kiekis, tačiau `switch()` elementas į save įtraukia visas galimybes, pagerina programą, be to prisiminkite, kad tik viena iš visų šių funkcijų bus atliekama.

Tai taip pat labai pagerina kodo skaitymą, nes `switch/case` konstrukcija yra lengvai skaitoma. Mes matysime tokias konstrukcijas ir toliau šioje knygoje, kurioje mes imsime reikšmę iš `HIDDEN` laukelio ir taip nustatysime tolimesnę skaičiavimų eigą. Kiekviena veiksmų eiga yra nurodoma funkcija, ir kai tik vartotojas pasirenka veiksmą – pvz. duomenų peržiūra ar jų keitimas `switch()` elementas pakeičia funkcija kuri ir turi atilikti norimus veiksmus.

Reikšmių sugražintų funkcijomis priskirimas kintamiesiems

Ženkime dar vieną žingsnį pirmyn. Funkcijos sugražina reikšmes, tad tas reikšmes galima priskirti kintamiesiems. Galite padaryti paprasčiausiai taip:

```
$HitCounter = number_of_hits();
```

Mūsų kintamasis `$HitCounter` gaus bet kokią reikšmę, kurią jam sugražins funkcija `number_of_hits()`. Tada jūs galite patalpinti kintamąjį tarp `switch()` elemento skliaustelių.

```
switch ($HitCounter)
  {
    case $HitCounter <100:
    echo "Not many hits this week";
    break;
    case $HitCounter <1000:
    echo "Some hits this week";
    break;
    case $HitCounter <10000:
    echo "Loads of hits this week";
    break;
  }
```

Ši funkcija bus vykdoma tik kartą ir priklausomai nuo rezultatų mes imsimės skirtingų veiksmų.

Reikšmių perdavimas

Mes jau apžvelgėme kaip funkcijos naudoja parametrus. Vienetinės reikšmės kurias mes pasiunčiame į funkcijas vadinamos **argumentais**. Skirtumas tarp argumentų ir parametrų yra tas, kad argumentus mes perduodame iškviesdami funkciją, o parametrai yra tai ką mes naudojame funkcijos viduje. Mes nenagrinėjome šio proceso plačiau, ir mums reikia apžvelgti šį klausimą plačiau, nes yra du būdai perduoti argumentus funkcijai ir priklausomai nuo to kuris buvo panaudotas, mes galime gauti skirtingus rezultatus.

Perdavimas per Reikšmę

Šį pirmąjį metodą mes jau naudojome. Mūsų mokesčių pavyzdys reikalavo perduoti į jį kontamąjį `$Alga`. Jie mes šiek tiek pakeisime programą, taip kad fiziškai pasikeistu kintamojo `$Alga` reikšmė, mes galėsime parodyti skirtumą tarp šių dviejų metodų.

```
function mokesciai ($Alga)
 {
  $Alga = $Alga - (($Alga/100)*20);
   return $Alga;
 }
$Alga = 2500;
echo (mokesciai($Alga)); // Tai pateiks 2000
echo $Alga; // Tai pateiks 2500
```

Tai niekaip nekeičia mūsų prieš tai daryto pavyzdžio, tačiau du pateikti atsakymai vis tiek bus skirtingi. Viskas ką mes padarėme yra perdavėme funkcijai kintamąjį kaip argumentą. Šiuo atvėju buvo perduota reikšmė 2500 ir funkcijos viduje, po atliktų veiksmų, buvo gautas naujas kintamasis, taip pat buvo pateikta kintamojo $Alga reišmė 2500:

```
function mokesciai (2500)
 {
  $Alga = $Alga - (($Alga/100)*20);
```

```
    return $Alga;
 }
$Alga = 2500;
echo (mokesciai($Alga)); // Tai pateiks 2000
echo $Alga; // Tai pateiks 2500
```

Šis procesas žinomas kaip argumento perdavimas **per reikšmę**. Ne svarbu kaip mes pakeisime reikšmę funkcijos viduje, reikšmė kurią mes suteikėme kintamajam `$Alga` išliks tokia pati, tad jei jūs pateksite `$Alga` turinį po funkcijos veikimo, ji vis tiek išliks 2500. Funkcija paprasčiausiai pateikia naujai apskaičiuotą reikšmę kuri yra 2000, ir daugiau nedaro nieko. Tai yra pagrindinis procesas kurį jūs naudosite pateikdami reikšmes funkcijai. Kas atsitiks jei jūs norėsite, kad kintamasis `$Alga` iš tikro turėtu naujai apskaičiuotą reikšmę?

Perdavimas per Referentą (referente)

Tai yra antrasis metodas kurio mes dar nenaudojome, kai perduota reikšmė iš tikro pakeičiama funkcijos viduje. Šis metodas vadinamas perdavimu **per referentą**. Norėdami nurodyti PHP, kad jūs norite naudoti kaip tik šį metodą jums reikia parašyti prieš kintamąjį **&** ženklą:

```
function mokesciai (&$Alga)
 {
  $Alga = $Alga - (($Alga/100)*20);
   return $Alga;
 }
$Alga = 2500;
echo (mokesciai($Alga)); // Tai pateiks 2000
echo $Alga; // Tai pateiks 2000
```

Šiuo atveju, kintamojo `$Alga` reikšmė yra pakeičiama, taip kad dabar šio kintamojo reikšmė bus 2000.

Numatytų reikšmių nustatymas

Kad dar laibiau viską sujaukti, jūs galite nurodyti jūsų parametrų reikšmes pačių argumentų viduje. Tai tampa numatytąją reikšme kai jūs nenurodote jokios kitos: Just to confuse matters, you can also set the values of your parameters within the arguments themselves. This
becomes the default value for when you don't specify any parameters:

```
function mokesciai (&$Alga = 2500)
 {
  $Alga = $Alga - (($Alga/100)*20);
   return $Alga;
 }
```

Tai reikšią, kad iškviečiant funkciją nenurodžius jokio argumento:

```
tax();
```

bus automatiškai panaudota reikšmė 2500. Tačiau jei jūs pateiksite argumenta pvz.:

```
tax(3000);
```

tada bus naudojama pateikta reikšmė, o ne numatytoji, tai yra dirbama su 3000, o ne su 2500.

Parametrų reikšmės

Jei jūs nenustatėte numatytosios reikšmės, ar neperdavėte funkcijai argumente PHP 4 pati priskirs argumentui nulinę reikšmę. Tai reiškia, kad jei

funkcijai turi būti pateikti du argumentai, o jūs perdavėte tik vieną, tada tas argumentas kurio jūs nepateikėte įgaus nulinę reikšmę. Jūs galite gauti įspėjantį pranešimą priklausomai nuo tokią PHP versiją jūs naudojate ir kaip sukonfigūruoti PHP pranešimai apie klaidas.

Mes paimsime pavyzdį kuriame funkcijai turi būti pateikti du argumentai:

```
function mokesciai($Alga, $Procentai)
 {
  return $Alga - (($Alga/100)*$Procentai);
 }
```

Dabar mes perduosime šiai funkcijai tik vieną argumentą:

```
echo (mokesciai (3000));
```

PHP tai interpretuos, kad reikšmė buvo suteikta tik pirmajam argumentui `$Alga`, ir pati suteiks nulinę reikšmę `$Procentai`. Kaip rezultatą funkcija pateiks tokią pačia reikšmę, nes dešinėje pusėje buvo apkaičiuotas nulis:

```
3000 – ((3000/100 * 0)
```

Tas pats principas pritaikomas ir dirbant su nustatytomis reikšmėmis, tad jei jūs parašysite pirmąjį parametrą su numatytąją reikšme:

```
function mokesciai($Alga=2500, $Procentai)
 {
  return $Alga - (($Alga/100)*$Procentai);
 }
```

ir taip pat ją iškviesite:

```
echo (tax(3000));
```

PHP supras, kad reikšmė 3000 yra perduodama `$Alga` ir nieko neperduodama kintamajam `$Procentai`, netgi jei `$Alga` turi numatytąją reikšmę ir jūs nieko nenorite jai perduoti.

Viso šito moralas tas – kad parametrai funkcijoje ir funkcijos iškvietimas turi sutapti, tiek savo kiekiu tiek vietomis, nebent jūs naudojate numatytąsias reikšmes. Jei taip ir yra prisiminkite parametrų seką. Šitas pavizdys nesukels jokių klaidų:

```
function mokesciai($Alga=2500, $Procentai)
 {
  return $Alga - (($Alga/100)*$Procentai);
 }
```

Tačiau jūs visada turėsite perduoti reikšmę kintamajam `$Alga` prieš perduodami ją `$Procentai`. Norint padaryti šį pavyzdį normaliai veikianti, jums tereikia parametrus sukeisti vietomis:

```
function mokesciai($Procentai, $Alga=2500)
 {
  return $Alga - (($Alga/100)*$Procentai);
 }
```

ir iškviesti funkciją štai taip:

```
echo(tax(25));
```

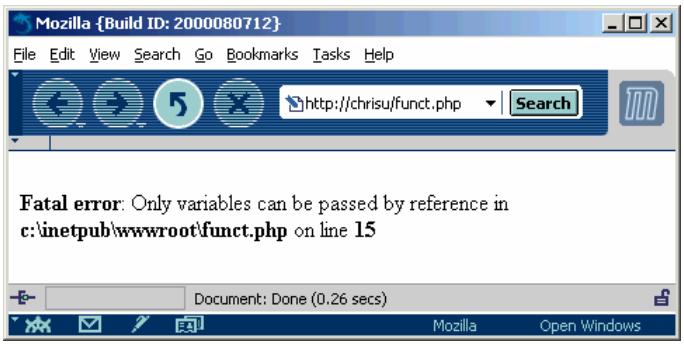
Parametrų su numatytomis reikšmėmis spąstai

Yra dar vienas dalykas į kurį mes turime atkreipti dėmesį naudodamiesi parametrais; šis funkcija iš pirmo žvilgsnio gali pasirodyti teisinga, tačiau iš tikro ji sugeneruos klaidą:

```
function mokesciai (&$Alga)
 {
```

```
  $Alga = $Alga - (($Alga/100)*20);
   return $Alga;
 }
echo (mokesciai(1000));
```

Jūs gausite štai tokį pranešimą, jei bandysite paleisti aukščiau pateiktą kodą:



Prieš tai buvusios PHP versijos, prieš 4'tają, leisdavo atlikti tokį veiksmą ir pranešdavo tik įspėjimą, tačiau PHP 4 baigė taip elgtis. Jei norite, kad kodas veiktu jums reikia pasiųsti kintamąjį, o ne skaitinę reikšmę, štai taip:

```
$Alga = 20;
echo (mokesciai(&$Alga));
```

Jūs negalite pasiųsti skaitinės reikšmės, nes neturėtu prasmės keisti skaitinę reikšmę. jei pasiųsite 1000, jūs nenorėsite, kad ši reikšmė pasikeistu. Ta pati taisyklė taikoma ir konstantoms.

Judėkime į priekį ir apžvelkime kitas funkcijų galimybes.

Kintamųjų rūšis

Šio skyriaus viduje mes užsiminėm, kad kintamieji esantys funkcijos viduje nebutinai egzistuoja ir už jos ribų. Tiesa pasakius dabar mes turime pristatyti visai naują koncepciją – kintamųjų **egzistavimo trukmė**. Kintamųjų egzistavimo trukmė yra laikotarpis nuo jo sukurimo iki egzistavimo pabaigos. Normaliai tai web puslapio egzistavimo laikotarpis. Tačiau kai kintamieji yra naudojami funkcijų viduje tai nevisada tiesa. Tai gali būti tik funkcijos egzistavimo laikotarpis. Funkcija yra iškviečiama ir kintamieji viduje pradeda egzistuoti. Jūs prieinate funkcijos galą, ir kintamieji uždaromi. Bet kokie kreipiniai į juos iš už funkcijos ribų yra klaidingi, nes šie kintamieji daugiau nebeegzistuoja.

Visuotiniai ir vietiniai kintamieji (global and local)

Kintamieji sukurti už funkcijos ribų egzistuoja per visą web puslapio egzistavimo laikotarpį. Visa koncepcija yra pavadinta apimtimi (**scope**). Kintamieji funkcijos viduje yra aprašomi kaip turintys **vietinę apimtį**, kai tuo tarpu kintamieji kurie egzistuoja per visą web puslapio egzistavimo laikotarpį vadinami **visuotinės apimties**.

Pažiurėkime į pavyzdį kuriame pateiktos abidvi šios kintamųjų rūšys. Šis kodas pateikia pasisveikinimą angliškai arba prancūziškai:

```
<?php
$WelcomeMessage = "Hello world";  Global Variable
```

```
function translate_greeting($WelcomeMessage)
 {
  $WelcomeMessage = "Bonjour Tout Le Monde";  Local
Variable
  return $WelcomeMessage;
 }
translate_greeting();
echo $WelcomeMessage;
?>
```

Pirmoji kintamojo `$WelcomeMessage` reikšmė yra "Hello World". Tai visuotinis kintamasis. Funkcijos viduje mes `$WelcomeMessage` suteikiame reikšmę "Bonjour Tout Le Monde", ir tada vygdome funkciją - `$WelcomeMessage` yra vietinis kintamasis. Jei įvygdisite šį kodą, puslapyje bus parašyta "Hello World". Taip yra nes funkcijos viduje esantis kintamasis yra vietinis.

Atkreipkite dėmesį į tai, kad mes čia nepateikiame funkcijos rezultatą, kaip kad darydavome anksčiau patalpindami funkciją į `echo()` komandą. Tačiau funkcija yra vygdoma, mes paprasčiausiai nepateikiame jos rezultato.

Jei mes pakeisme argumentą funkcijos parametruose ir kintamąjį pačioje funkcijoje ir vėl įvykdysime programą mes negausime jokio rezultato:

```
<?php
$WelcomeMessage = "Hello world";
function translate_greeting($FrenchMessage)
 {
  $FrenchMessage = "Bonjour Tout Le Monde";
  return $FrenchMessage;
 }
translate_greeting();
echo $WelcomeMessage;
echo $FrenchMessage;
?>
```

Taip yra, todėl, kad kintamojo egzistavimas yra nutraukiamas kai tik funkcija pateikia atsakymą.

Visuotinių kintamųjų naudojimas funkcijose

Analogiškai, jei mes norėtumėme pavaizduoti `$WelcomeMessage` turinį funkcijos viduje, mes negalėtumėme to padaryti nes `$WelcomeMessage` neegzistuoja funkcijos viduje:

```
<?php
$WelcomeMessage = "Hello world";
function translate_greeting($FrenchMessage)
 {
  echo $WelcomeMessage;
  $FrenchMessage = "Bonjour Tout Le Monde";
  return $FrenchMessage;
 }
translate_greeting();
echo $WelcomeMessage;
echo $FrenchMessage;
?>
```

Tai turi savo prasme – kas atsirtiktu jei mes turėtumėme vietinį kintamąjį funkcijos viduje tokiu pačiu vardu kaip ir globalus kintamasis už jos ribų? Mes turi turėti kažkokį būdą identifikuoti visuotinius kintamuosius, nes turi būti tikri kad PHP juos supranta kaip tuos pačius kintamuosius, o ne tik besidalinnačius vienodais vardais. Yra du būdai tai padaryti. Pirmasis parodytas žemiau:

```
<?php
$WelcomeMessage = "Hello World";
function translate_greeting($FrenchMessage)
 {
```

```
  global $WelcomeMessage;
  echo $WelcomeMessage;
  $FrenchMessage = "Bonjour Tout Le Monde";
  return;
 }
translate_greeting();
echo $WelcomeMessage;
?>
```

Šiame pavyzdyje, mes nurodėme, kad `$WelcomeMessage` yra globalus kintamasis. Kaip matėme anksčiau, jei mes pavaizduosime `$WelcomeMessage` funkcijos viduje prieš tai nenurode jo kaip globalaus kintamojo, jis bus traktuojamas kaip naujai sukurtas kintamasis ir neturės jokios reikšmės. Dabar gi mūsų `echo()` komanda parašo "Hello World" kaip ir buvo tikėtasi nes kintamasis taro globalios apimties.

Kitas būdas kaip pasiekti tokį pat rezultatą yra PHP kalboje naudoti `$GLOBALS` masyvą. Norit pavaizduoti globalų kintamąjį funkcijos viduje pasinaudojus `$GLOBALS` masyvu jums reikia:

```
<?php
$WelcomeMessage = "Hello world";
function translate_greeting($FrenchMessage)
  {
   echo $GLOBALS["WelcomeMessage"];
   $FrenchMessage = "Bonjour Tout Le Monde";
   return $FrenchMessage;
  }
translate_greeting();
echo $WelcomeMessage;
echo $FrenchMessage;
?>
```

Abiejuose pavyzdžiuose jūs gausite du kartus parašytą pranešimą "Hello World". Tai yra nes mes parašome jį vieną kartą funkcijos viduje ir kitą už funkcijos ribų.

Raktažodį `global` gana lengva suprasti, tačiau su `$GLOBALS` masyvu kiek sudėtingiau. Pažvelkime kiek atidžiau kaip šis masyvas veikia. Jis turi sekantį formatą:

```
$GLOBALS["KintamojoVardas"]
```

Norėdami apibūdinti kintamąjį jūs parašote jo vardą be `$` simbolio tarp kabučių ir lauštinių skliaustų. Prieš visą tai rašosi `$GLOBALS` didžiuosiomis raidėmis. Tiesa pasakius jis veikia taip pat kaip ir eilutės tipo masyvas kurį mes aptarėme ankstesneme skyriuje.

Vietinių kintamųjų reikšmių išlaikymas

Kas atsitinka kai mes iškviečiame funkcija vėl ir vėl. Kaip mes jau sakėme vietiniai kintamieji atsiranda kai funkcija yra iškviečiama ir nustoja egzistuoti kai ji baigia savo darbą. Tačiau gali būti tokių situacijų kai jūs norėsite, kad vietinių kintamųjų reikšmės išliktu tarp funkcijos iškvietimų. Jūs galėsite iškviesti funkciją dar kartą ir ji operuos reikšmėmis išsaugotomis nuo ankstesniojo karto. Įsivaizduokite, kad jūs atliekate kažkokius pasttovius skaičiavimus, kuriuos atlieka funkcija. Bus negerai jei ji kiekviena kartą pradės dirbti su naujais kintamaisiais.

```
function number_of_hits_on_web_site()
 {
  return $number_of_people = $number_of_people+1;
 }
```

Kintamieji kurie egzistuoja tarp funkcijos iškvietimų vadinami statiniais. Norėdami apibrėžti tokį kintamąjį jūs turite parašyti raktažodį static prieš kintamąjį:

```
function number_of_hits_on_web_site()
 {
  static $number_of_people = 0;
  return $number_of_people = $number_of_people+1;
 }
```

Tai neatrodo visai logiškai. Tiesa pasakius kiekviena kartą kai iškviečiama funkcija kintamojo reikšmė vis tiek yra nulis. Static reikšmė tame, kad linija su `static` apibrėžtimi yra atliekama tik vieną kartą pirmą kartą iškvietus funkciją, kitais kartais ji yra praleidžiama.

Jei jūs parašysite kiekvieną kartą funkcijos pateikiamą reikšmę, ji didės vienu skaičiumi:

```
echo(number_of_hits_on_web_site()); <-- would return 1
echo(number_of_hits_on_web_site()); <-- would return 2
echo(number_of_hits_on_web_site()); <-- would return 3
```

Mes apžvelgėme globalius, vietinius ir statinius kintamuosius – pakartokime pagrindinius jų skirtumus:

- Globalūs kintamieji turi reikšmes kurios egzistuoja visą programos veikimo laiką, tačiau norint juos panaudoti funkcijos viduje jūs turite juos apibrėžti su `GLOBALS` raktažodžiu.

- Vietiniai kintamieji turi reikšmes kurios egzistuoja tik funkcijos viduje ir tik tą laikotarpį kol vykdoma funkcija.

- Statiniai kintamieji yra vietiniai kintamieji kurie išlaiko savo reikšmę funkcijos viduje kievieną kartą kai funkcija yra iškviečiama.

Jei supratote šiuos skirtumus, pats laikas atlikti nedidelį pavyzdį. Mes sukursime kikevienos rūšies kintamuosius ir pavaizduosime juos web puslapyje, nurodydami kada mes esame funkcijos viduje, o kad už jos ribų.

Išbandykite – Skirtingų rūšių naudojimas

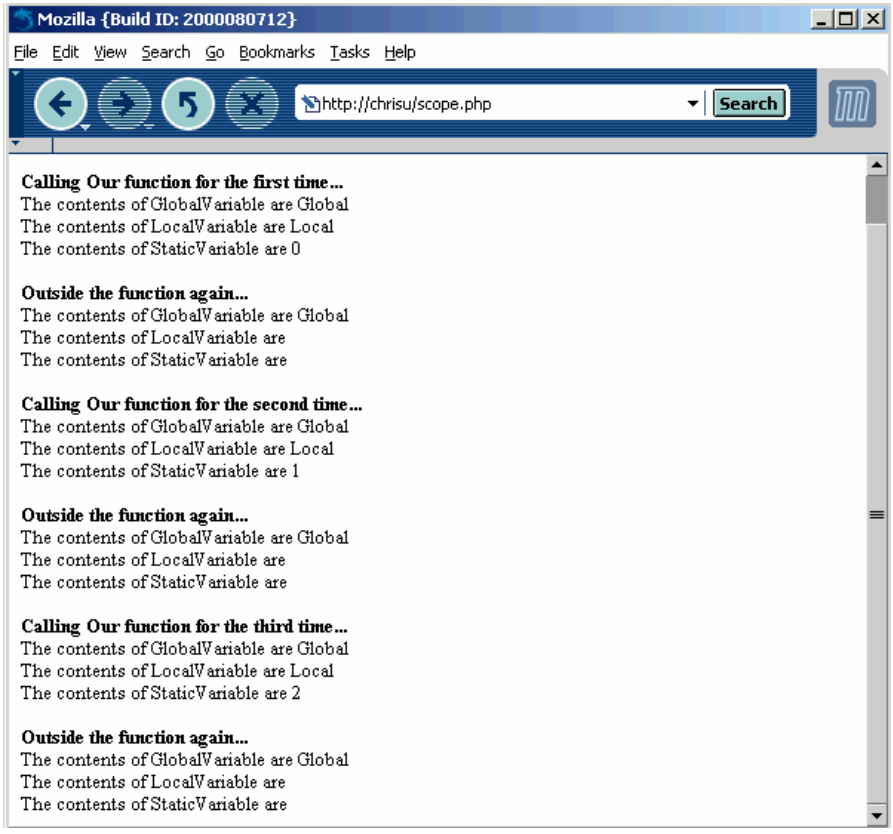
1. Atidarykite savo teksto redaktorių ir parašykite:

```
<HTML>
<HEAD></HEAD>
<BODY>
<BR>
<BR>
<FONT SIZE=-1>
<?php
$GlobalVariable = "Global";
function local()
 {
  $LocalVariable="Local";
  static $StaticVariable=0;
  echo "<BR>The contents of GlobalVariable are " .
$GLOBALS["GlobalVariable"];
  echo "<BR>The contents of LocalVariable are
$LocalVariable";
  echo "<BR>The contents of StaticVariable are
$StaticVariable";
  return $StaticVariable=$StaticVariable+1;
 }
echo "<B>Calling Our function for the first time...</B>";
local();
echo "<BR><BR><B>Outside the function again...</B>";
echo "<BR>The contents of GlobalVariable are
$GlobalVariable";
echo "<BR>The contents of LocalVariable are
$LocalVariable";
```

```
  echo "<BR>The contents of StaticVariable are
$StaticVariable";
  echo "<BR><BR><B>Calling Our function for the second
time...</B>";
  local();
  echo "<BR><BR><B>Outside the function again...</B>";
  echo "<BR>The contents of GlobalVariable are
$GlobalVariable";
  echo "<BR>The contents of LocalVariable are
$LocalVariable";
  echo "<BR>The contents of StaticVariable are
$StaticVariable";
  echo "<BR><BR><B>Calling Our function for the third
time...</B>";
  local();
  echo "<BR><BR><B>Outside the function again...</B>";
  echo "<BR>The contents of GlobalVariable are
$GlobalVariable";
  echo "<BR>The contents of LocalVariable are
$LocalVariable";
  echo "<BR>The contents of StaticVariable are
$StaticVariable";
  ?>
  </FONT>
  </BODY>
  </HTML>
```

2. Išsaugokite kaip `scope.php`.

3. Atidarykite savo naršyklėje `scope.php`:



Kaip tai veikia

Ši programa neturi praktinės reikšmės, tačiau skirtingos kintamųjų rūšys yra sudėtindas dalykas ir mes norėjome kiek galima išsamiau paaiškinti jo prigimtį. Pirmoji mūsų programos eilutė sukuria kintamąjį `$GlobalVariable` ir priskiria jam reikšmę:

```
$GlobalVariable = "Global";
```

Tada mes parašome mūsų funkciją:

```
function local()
 {
```

Viduje mes sukuriame vietinį kintamąjį `$LocalVariable`, ir statinį kintamąjį `$StaticVariable`:

```
$LocalVariable="Local";
static $StaticVariable=0;
```

Sekančios trys linijos parašo globalaus, vietinio ir statinio kintamojo reikšmes funkcijoje:

```
  echo "<BR>The contents of GlobalVariable are " .
$GLOBALS["GlobalVariable"];
  echo "<BR>The contents of LocalVariable are
$LocalVariable";
  echo "<BR>The contents of StaticVariable are
$StaticVariable";
```

Galiausiai, `return` komanda didina statinio kintamojo reikšmę vienetu:

```
return $StaticVariable=$StaticVariable+1;
}
```

Mūsų programa prasideda `local()` funkcijos iškvietimu:

```
echo "<B>Calling Our function for the first time...</B>";
local();
```

Mes matome globalaus, vietinio ir statinio kintamųjų reikšmes kurios yra "Global", "Local", ir 0 funkcijos viduje. Kai funkcija yra pabaigiama, mes parašome šių trijų kintamųjų reikšmes kokios jos yra už funkcijos ribų:

```
  echo "<BR><BR><B>Outside the function again...</B>";
  echo "<BR>The contents of GlobalVariable are
$GlobalVariable";
  echo "<BR>The contents of LocalVariable are
$LocalVariable";
  echo "<BR>The contents of StaticVariable are
$StaticVariable";
```

Už funkcijos, kaip ir galima tikėtis vietinis ir statinis kintamasis yra tušti ir vienintelis `$GlobalVariable` turi reikšmę. Toliau, mes iškviečiame funkciją dar kartą:

```
  echo "<BR><BR><B>Calling Our function for the second
time...</B>";
  local();
```

Dar kartą `$GlobalVariable` yra lygus "Global", `$LocalVariable` yra inicijuojama iš naujo ir turi tokią pačią reikšmę, tačiau `$StaticVariable` reikšmė yra kitokia, nes prieš tai buvusį kartą mūsų funkciją ją padidino vienetu. Šį kartą parašomas 1. Mes vėl einame už funkcijos ribų ir vėlgi tik `$GlobalVariable` turi reikšmę:

```
  echo "<BR><BR><B>Outside the function again...</B>";
  echo "<BR>The contents of GlobalVariable are
$GlobalVariable";
```

```
  echo "<BR>The contents of LocalVariable are
$LocalVariable";
  echo "<BR>The contents of StaticVariable are
$StaticVariable";
```

Galiausiai, mes iškviečiame funkciją trečiąjį kartą ir šį kartą vienintelis skirtumas yra tas kad statinis kintamasis padidėjo dar vienu vienetu:

```
  echo "<BR><BR><B>Calling Our function for the third
time...</B>";
  local();
```

Už funkcijos ribų, kaip tikriausiai ir spėjote niekas nepasikeitė. Laimei tai užbaigia mūsų programą ir mes nesiruošiame toliau nagrinėti šios temos. Jei jūs gavote kitokius atsakymus patikrinkite savo kodą, jūs turite jau patys sugebėti rasti klaidas.

Toliau neturiu nei laiko nei entuziazmo versti šią knygą. Jei kas pratęs šį darbą, kuo aš beje labai abejoju, tik sveikinsiu.

Programuotojas programuotojui





Pradedant PHP4

PHP yra greitai auganti Web technologija leidžianti programuotojams daryti dinaminius, interaktyvias programas, įterpti duomenis iš duomenų bazių, bei pasinaudoti tokiomis savybėmis kaip e-mail integracija ar dinamiškai generuojami paveiksliukai.
PHP4 pristatė daugybe naujovių, padarydama web programavimą dar lengvesniu ir ši knyga pademonstruos kaip pasinaudoti visais programavimo kalbos privalumais

Ši knyga yra išsamus PHP kalbos vadovas, pradedant pagrindais ir prieinant iki pilnai interaktyvių puslapių kūrimo. Pilnai veikiantis šios knygos pavyzdžiai parodo paieškos sistemos, pašto administravimo sistemas, internatini failu administravimą, bei grafinę internetinę parduotuvę.

Kam skirta ši knyga?

Visi kas susipažinę su HTML gali naudotis šia knyga. Jei jau anksčiau programavote jums bus lengviau, bet tai neprivaloma. Visus pateiktus kodus jūs galite vykdyti tiek Windows tiek ir Linux sistemose, o knyga paaiškina kaip konfiguruoti web serverį ir duomenų bazę

Kas yra šioje knygoje?

- Pilną PHP kalbos vadovas
- Instaliacijos ir problemų sprendimo instrukcijas
- Pristatomos duomenų bazės ir konkrečiai MySQL
- Visas kodas veikia tiek Windows tiek ir Linux OS
- Pridėtas detalus programavimo kalbos žinynas

Norite padėti?
Verskite knygas, nelaukite kol kažkas atneš jums jas ant lėkštutės.

Nebūkite abejingi, nors tokie tikriausiai ir esate.

Web puslapis - xxxxxxxx
Email - xxxxxxxx
Forumas - xxxxxxxx

Visa knygos medžiaga yra vertimas iš originalo. Vertėjai neatsako už visas galimas klaidas nesklandumus ir nesusipratimus.

Mes niekaip nesame susiję su Wrox ar jų atstovais.

Mūsų tikslas supažindinti jus su PHP ir suteikti jums galimybę patiems kurti ir mokyti kitus

Free eBooks

$39.99 USA
$59.99 CANADA
$00.00 LITHUANIA

Rekomenduojama knygų grupė
PHP