PHP-KIELEN PERUSRAKENTEET

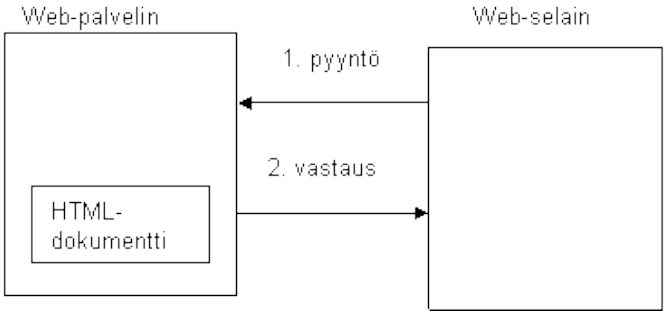
1 PHP-sovelluksen toiminta

Web-sivujen toiminta perustuu asiakas-palvelin malliin:

- Asiakas(selain) lähettää palvelimelle pyyntöjä
- Palvelinohjelma vastaa asiakasohjelman pyyntöihin
- Protokolla eli yhteyskäytäntö määrittelee miten asiakasohjelma ja palvelinohjelma viestivät keskenään.

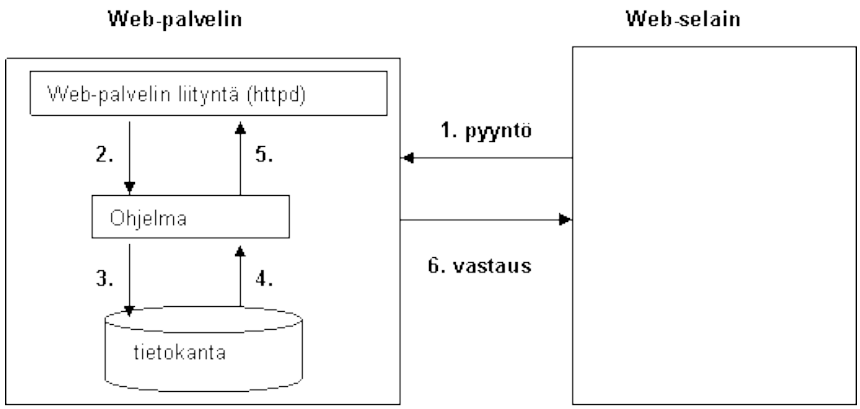
Staattinen Web-dokumentti

Staattinen eli pysyvä web-dokumentti on muuttumaton siten, että vain dokumentin tekijä voi sitä muuttaa. Käyttäjä voi vaikuttaa esitettyyn tietoon lähinnä hyperlinkkien avulla. Staattiset dokumentit sopivat tilanteisiin, jossa vuorovaikutusta ei tarvita eikä tarvita päivitystä usein.



Dynaaminen Web-dokumentti

Dynaaminen Web-dokumentti sisältää muuttuvaa, esimerkiksi tietokannasta haettavaa, tietoa. Dokumentti voidaan luoda automaattisesti kulloisenkin tilanteen pohjalta. Dynaaminen web-dokumentti vaatii yleensä ohjelmointia jossain määrin.



Web-sovellusten toteuttamistekniikoita

Asiakastekniikat eli selaimessa toteutettavat tekniikoita:

- HTML (lomakkeet)
- CSS
- JavaScript ja muut selainten scriptikielet
- DHTML
- Java-sovelmat (appletit)
- ActiveX, Flash,…plug-in-ohjelmat

Palvelintekniikoita:

- Common Gateway Interface (CGI)
- Java Servletit
- Upotetut tekniikat: PHP, asp, JSP,..
- Sovelluspalvelimet

2 PHP-scripti

PHP-scripti näyttää muuten tavalliselta HTML-dokumentilta, mutta HTML-komentojen sekaan on upotettu PHP-kielisiä komentoja.

PHP-koodi erotetaan HTML-koodista tagein <? ja ?> tai <?php ja ?>

PHP-komento päättyy aina puolipisteeseen (;).

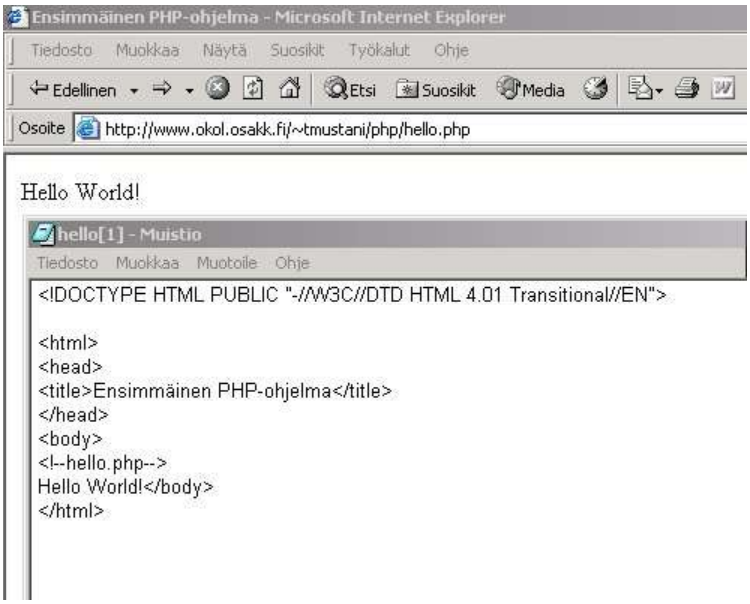
Tiedoston tallennusmuodoksi kirjoitetaan useimmiten .php. Tiedostotunnisteen perusteella palvelin päättelee, että tiedosto sisältää PHP-koodia, jonka palvelin suorittaa ennen dokumentin lähettämistä selaimelle. Palvelimen asetuksista riippuu mitkä tiedostotunnisteet tulkitaan PHP-tiedostoiksi. Palvelin voi tunnistaa PHP-tiedostoksi esimerkiksi .php, .phtml ja .php3 -tiedostot.

Esimerkki: Tehdään ohjelmointikirjoista tuttu Hello World! -ohjelma. Kirjoitetaan alla oleva koodi:

```
<html>
<head>
  <title>Ensimmäinen PHP-ohjelma</title>
</head>
<body>
<!-- hello.php-->
<?php
  echo "Hello World!";
?>
</body>
</html>
```

Tallennetaan tiedosto nimellä hello.php ja siirretään se palvelimelle. Tiedostoa ei voi esikatsella työaseman selaimella ennen sivun siirtämistä palvelimelle, koska palvelinohjelmiston pitää ensin suorittaa PHP-koodi.

Selaimella katsottaessa sivun tulisi näyttää alla olevan kuvan mukaiselta. Katsottaessa sivun lähdekoodia selaimen kautta, ei PHP-koodia näytetä, koska palvelin on suorittanut koodin ja lähettänyt selaimelle vain HTML-osan.



Huom! Useimmissa kurssin koodiesimerkeissä on tämän jälkeen vain PHP-osa. Sivulle aina tulevat <html>, <head> ja <body> tagit on jätetty esimerkeistä pois.

3 Kommentit ja sivulle tulostaminen

Koodia voidaan kommentoida kolmella eri tavalla:
// yksirivinen kommentti, rivin loppuosa on kommenttia
myös tämä on yksirivinen kommentti, rivin loppuosa on kommenttia
/* monirivinen
kommentti */

PHP-moodista tulostetaan tekstiä palautettavalle HTML-sivulle joko echo-komennolla tai print-funktiolla. Jos tulostettavaa tekstiä halutaan muotoilla siirtymättä välillä HTML-moodiin, tulee myös muotoiluun käytettävät HTML-tagit sijoittaa tulostuslauseeseen.

Esimerkki: Molemmat tulostavat saman teksti HTML-sivulle

```
echo "<h1>Suurempi teksti!";
print "<h1>Suurempi teksti!";
```

Rivinvaihto saadaan aikaan \n -merkillä.

4 Muuttujat ja vakiot

Muuttujia ei tarvitse esitellä PHP-kielessä. Muuttujan tyyppi määräytyy automaattisesti sen mukaan minkä tyyppistä tietoa siihen sijoitetaan. Vasta tämän perusteella varataan muistista muuttujan vaatima tila.

Muuttujan nimi alkaa aina dollarimerkillä ($). Nimissä voi käyttää kirjaimia numeroita ja alaviivaa. Muuttujan nimi ei kuitenkaan voi alkaa numerolla. Isot ja pienet kirjaimet huomioidaan muuttujan nimessä.

$Nimi ja $nimi ovat kaksi eri muuttujaa.

Esimerkki:

```
$nimi="matti";
$ika=32;
```

Vakioihin tallentaan numeerista tai tekstipohjaista tietoa, joka säilyy muuttumattomana ohjelman suorituksen ajan. Niitä voidaan hyödyntää esimerkiksi tulostettaessa samanlaisina toistuvia tekstejä. Vakio määritellään define-avainsanalla.

Esimerkki:

```
define("PII", 3.14159);
define("TEKSTI", "Tulos on: ");
$sade=2.5;
$ala=PII*$sade*$sade;
echo TEKSTI .$ala;
```

5 Tietotyypit

Yksinkertaiset tietotyypit:

- boolean: totuusarvo, saa arvot true ja false
- integer: kokonaisluku välillä -2 147 483 - 2 147 483 647 Jos ylitetään rajat, niin PHP muuttaa luvun automaattisesti float-tyyppiin.
- float: liukuluku, "desimaaliluku"
- string: merkkijono Muuttujaan sijoitettava merkkijono tulee olla lainausmerkkien tai heittomerkkien sisällä.

Rakenteiset tietotyypit:

- array: taulukko
- object: olio

Muuttujan tietotyyppi saadaan selville funktiolla gettype():

```
echo gettype($nimi);
```

Muuttujan tietotyyppi voidaan muuttaa funtiolla settype():

```
settype($numero,"string");
```

Tulostettaessa tekstiä lainausmerkkien sisään ei muuttujia tarvitse erikseen erottaa, vaan ne tunnistetaan $-merkistä. Heittomerkeillä esitetyssä merkkijonossa olevat muuttujat tulostuvat sellaisenaan. Muuttujia ja tekstiä tai funktioita voidaan yhdistää pisteellä.

```
$ika=20;
$nimi="Pekka";
print "$nimi ikää $ika vuotta";
print $nimi ." ikää " .$ika ." vuotta"; //sama lopputulos kuin edellä
print '$nimi ikää.$ika vuotta'; //Ei toimi oikein
print $nimi .' ikää' .$ika .'vuotta';//Pitää tehdä näin käytettäessä heittomerkkejä
```

Esimerkki:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Muuttujakokeilu</title>
</head>
<body>
<!--muuttujat.php-->
<h1>Kokeillaan php-muuttujia</h1>
<?php
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Muuttujakokeilu</title>
</head>
<body>
<!--muuttujat.php-->
<h1>Kokeillaan php-muuttujia</h1>
<?php
  //Sijoitetaan muuttujiin eri tyyppisiä arvoja
  $nimi="Maija";
            $ika=18;
            $pituus=1.72;
            $opiskelija=true;

            //tulostetaan muuttujien arvot
            print "<h2>Henkilötiedot:</h2>";
            print "Nimesi: $nimi <br>" ;
            print "Ikäsi: $ika <br>";
            print "Pituutesi: $pituus<br>";
            print "Opiskelija: $opiskelija";

            /*tulostetaan muuttujien tietotyypit gettype-funktiolla
            teksti ja muuttuja yhdistetään pisteellä tulostuslauseessa*/
            print "<h2>Muuttujat ja tietotyypit:</h2>";
            print "nimi: ".gettype($nimi) ."<br>";
            print "ikä: ".gettype($ika)."<br>";
            print "pituus: ".gettype($pituus)."<br>";
            print "opiskelija: ".gettype($opiskelija)."<br>";
           ?>
</body>
</html>
```

6 Operaattorit

Aritmeettiset operaattorit

+, -, *, /	peruslaskutoimitukset
%	jakojäännös
++	inkrementointi eli muuttujan arvon kasvatus yhdellä
--	dekrementointi eli muuttujan arvon vähentäminen yhdellä
.=	liittäminen merkkijonoon

Jos ++ tai -- on muuttujan vasemmalla puolella suoritetaan ennen muuttujan arvon tulkitsemista. Jos merkki on muuttujan oikealla puolella, suoritetaan operaatio ensin ja kasvatetaan muuttujan arvoa vasta sitten.

Sijoitusoperaattorit

=	sijoitus
+=	lisää ja sijoita (muut peruslaskutoimitukset vastaavasti)

Merkkijonoja liitetään toisiinsa . operaattorilla: "Matti" . " Meikäläinen"
Esimerkki: Mieti ensin ja kokeile sitten, päättelitkö oikein mitä seuraava ohjelma tulostaa.

```
<?php
 $luku=0;
   $luku++;
   $luku+=8;
   $luku--;
   $luku-=3;
   $jakoj=$luku%2;
   print "Luku on " .$luku;
   print"<br>";
   print "Kun luku jaetaan 2:lla on jakojäännös" .$jakoj;
 ?>
```

Vertailuoperaattorit

= =	yhtä suuri kuin
<	pienempi kuin (suurempi kuin vastaavasti)
< =	pienempi tai yhtäsuuri kuin (suurempi kuin vastaavasti)
<>	erisuuri kuin
!=	erisuuri kuin
!= =	erisuuri kuin tai eivät samaa tyyppiä

Huom. = on sijoitus operaattori, verrattaessa yhtäsuuruutta käytetään == operaattoria!
Loogiset operaattorit

AND tai &&	ja
OR tai ||	tai
XOR	poissulkeva tai
!	ei

7 Ohjausrakenteet

7.1 Valintarakenne: If-lause

if-rakennetta voidaan käyttää yksittäiseen- tai monivalintatilanteeseen. Ehtolauseke esitetään suluissa. Suoritettavat lauseet kootaan yhdeksi lohkoksi lohkosululla { }. Lauseet suoritetaan, mikäli ehto saa arvon true eli tosi.

Esimerkki: Tulostetaan virheilmoitus, mikäli pistemäärä on virheellinen.

```
if($pisteet>35)
{
   print "Pistemäärä ei voi olla yli 35, syötä uudelleen!";
}
```

Esimerkki: Tulostetaan arvosana pistemäärän perusteella. Mieti, mitä ohjelma tulostaisi.

```
<?php
  $pisteet=25.5;
  if($pisteet < 10)
                {
                  $tulos="hylätty";
                }
                else if($pisteet<15)
                {
                  $tulos="T1";
                }
                else if($pisteet<20)
                {
                  $tulos="T2";
                }
                else if($pisteet<25)
                {
                  $tulos="H3";
                }
                else if($pisteet<30)
                {
                  $tulos="H4";
                }
                else
                {
                  $tulos="K5";
                }
                echo $tulos;
?>
```

else if -osia voi olla useampia tai se voi kokonaan puuttua. Myös else -osa voi puuttua.

7.2 Valintarakenne: Switch-lause

Switch-lauseessa suoritetaan ensin suluissa oleva lauseke. Sen arvon perusteella siirrytään siihen case-haaraan, jonka vakiolauseke vastaa laskettua arvoa. Case-haarassa olevat lauseet suoritetaan break-lauseeseen saakka, jonka jälkeen poistutaan Switch-lauseesta.

Switch-lauseessa voi olla default-osa, johon siirrytään mikäli yksikään case-haaroista ei vastaa laskettua arvoa. Default-osa voi myös puuttua.
Esimerkki: Alennusprosentti valitaan ammatin perusteella. Mieti, mitä ohjelma tulostaisi.

```
<?php
$ammatti="elake";
switch($ammatti)
{
  case "opiskelija":
    $ale=10;
    break;
  case "elake":
    $ale=20;
    break;
  case "varusmies":
    $ale=15;
    break;
  default:  // suoritetaan, jos mikään ehto ei tosi
    $ale=0;
    break;
}
  echo $ale ."%"
?>
```

7.3 Toistorakenne: While-lause

While rakenne voi olla alku- tai loppuehtoinen toisto. Alkuehtoisessa toistossa eli While-rakenteessa toistettavia lauseita ei suoriteta kertaakaan, jos ehto ei ole tosi.

Loppuehtoisessa toistossa eli do..while -rakenteessa lauseet suoritetaan ainakin kerran.

Seuraavassa on esimerkki molemmista tavoista samassa ohjelmassa.

Esimerkki 1: Alkuehtoinen toisto. Mieti, mitä ohjelma tulostaa.

```
<?php
  $rahat=200;
  define ("OSTOS",250);  //määritellään vakio OSTOS
  while ($rahat-OSTOS > 0)
  {
    $rahat=$rahat-OSTOS;
    echo "While-rakenne: Jäi vielä $rahat euroa \n"; //\n on rivin vaihto
  }
?>
```

Esimerkki 2: Loppuehtoinen toisto. Mieti, mitä ohjelma tulostaa.

```
<?php
  $rahat=200;
  do
  {
    $rahat=$rahat-OSTOS;
    echo " Do while-rakenne: Jäi vielä $rahat euroa \n";
  }
  while ($rahat-OSTOS > 0);
?>
```

7.4 Toistorakenne: For-lause

For-lausetta käytetään, kun tiedetään ennkkoon, kuinka monta kertaa toisto suoritetaan. For-lauseen syntaksi on:

```
for (lauseke1;lauseke2;lauseke3)
  {
    toistettavat lauseet;
  }
```

- Lauseke 1 suoritetaan vain kerran silmukan alussa. Asetetaan muuttujan alkuarvo.
- Lauseke 2 suoritetaan jokaisen toistokierroksen aluksi. Jos lausekkeessa oleva ehto on true, toistettavat lauseet suoritetaan.
- Lauseke 3 suoritetaan jokaisen kierroksen lopussa. Kasvatetaan muuttujan arvoa.

Esimerkki:Mieti, mitä ja miten seuraava ohjelma tulostaa.

```
<?php
  echo "Luvut 1-10:";
  for ($i=1; $i<=10; $i++)
  {
    echo "<br>$i";
  }
?>
```

8 Taulukot

Taulukkomuuttuja tunnistetaan muuttujan nimen perässä olevista hakasuluista:

```
$taulukko[]
```

Taulukon kokoa ei tarvitse määritellä etukäteen. Taulukko on dynaaminen eli se varaa tilaa muistista dynaamisesti tarvittaessa.

Taulukon voi luoda sijoittamalla arvoja taulukkomuuttujaan. Jos sijoituslauseessa ei ole indeksiä, niin sijoitettavan arvon indeksi on yhtä suurempi kuin suurin käytössä oleva kokonaislukuindeksi:

```
$nimet[]="Pekka";  //$nimet[0]="Pekka
  $nimet[]="Paavo";   //$nimet[1]="Paavo"
  $nimet[5]="Paavali"; // $nimet[5]="Paavali"
  $nimet[]="Pauli";   //$nimet[6]="Pauli"
```

Taulukon voi luoda myös array-käskyllä:

```
$lapset=array(0=>"Pekka",1=>"Paavo",5=>"Paavali","Pauli");
  $pojat=array("Pekka","Paavo","Paavali","Pauli");
```

Taulukko voidaan tulostaa avaimineen funktiolla print_r($taulukko) ;

Numeeriset indeksit omaava taulukko voidaan käydä läpi foreach() silmukalla:

```
foreach($nimet as $val)
  {
    echo "$val <br>";
  }
```

Vastaavasti assosiatiivinen taulukko käydään läpi samalla rakenteella:

```
foreach($henkilo as $key=>$value)
  {
    echo "$value <br>";
  }
```

Taulukko voidaan lajitella funktiolla sort() ja assosiatiivinen taulukko funktiolla asort(). Lajittelu voidaan tehdä myös avaimen mukaan funktiolla ksort():

```
 sort($nimet);
  asort($henkilo);
  ksort($henkilo);
  krsort($henkilo);   //käänteinen järjestys
  rsort($nimet);  //käänteinen järjestys
```

Esimerkki: Yhdistetään edelliset esimerkit yhteen ohjelmaan. Mieti, mitä eroa eri taulukoilla on ja mitä ohjelma tulostaa. Kokeile vasta sitten.

```php
<?php
  $nimet[]="Pekka";
  $nimet[]="Paavo";
  $nimet[5]="Paavali";
  $nimet[]="Pauli";

 //tulostetaan avaimet ja taulukon sisältö
 print "Taulukko nimet:<br>";
 print_r($nimet);

 print("<p>");
 $lapset=array(0=>"Pekka",1=>"Paavo",5=>"Paavali","Pauli");
 $pojat=array("Pekka","Paavo","Paavali","Pauli");

 //Tulostetaan taulukot avaimineen
 print "Taulukko lapset:<br>";
 print_r($lapset);
 print("<p>");
 print "Taulukko pojat<br>";
 print_r($pojat);

 print "<p>";
 $henkilo=array(nimi=>"Pekka", osoite=>"Keskuskatu 5", puh=>"040-0400400");
 $henkilo[ika]="36";
 //Tulostetaan assosiatiivinen taulukko
 print("<br> Assosiatiivinen taulukko: <BR>");
 print_r($henkilo);
 print("<BR>");
 print "<p>";
 print("Tulostus foreach-rakenteella nimet-taulukko lajiteltuna:<br>");
 sort($nimet);
 foreach($nimet as $value)
 {
   echo "$value <br>";
 }
 print "<p>";
 print("Tulostus foreach-rakenteella assosiatiivinen henkilo-taulukko lajiteltuna indeksin mukaan:<br>");
 ksort($henkilo); //lajitellaan assosiatiivinen taulukko
 foreach($henkilo  as $key=>$value)
 {
 echo "$key on $value <br>";
 }

?>
```

9 Funktiot

Funktiot eli aliohjelmat ovat pieniä ohjelman osia, jotka suorittavat jonkin rajatun tehtävän. PHP:n sisäiset funktiot eli kirjastofunktiot ovat kaikkien ohjelmien käytössä. Niiden dokumentaation voi tarkistaa esim. http://www.php.net/ . Olemme jo käyttäneet edellisissä esimerkeissä muutamaa sisäistä funktiota kuten print() ja sort().

Käyttäjän itse määrittelemän funktion rakenne on:

```php
 function funktion_nimi($muodollinen_parametri1,$muodollinen_parametri2,..)
 {
   lauseet;
   return $paluuarvo;
 }
```

Esimerkkejä: Yksinkertaisin funktio ilman parametrejä:

```
<?php
  function tervehdi()
  {
    print "<h1> Hello World! </h1>";
  }

  tervehdi(); //kutsutaan pääohjelmassa funktiota tervehdi
?>
```

Parametrien välitys funktiolle:

```
<?php
  function tervehdys($nimi)
  {
    print "<h1> Hello $nimi! </h1>";
  }

  tervehdys("Mike"); //kutsutaan pääohjelmassa funktiota tervehdys
?>
```

Arvon palauttaminen:

```
<?php
  function laske($luku1,$luku2)
  {
    $summa=$luku1+$luku2;
    return $summa;
  }

  echo "Lukujen summa on " .laske(3,4); //kutsutaan pääohjelmassa funktiota laske
?>
```

Teksti ja funktion kutsu erotetaan tulostuslauseella pisteellä.

Globaali ja lokaali muuttuja

Normaalisti muuttuja näkyy vain siinä ohjelmalohkossa, jossa se on määritelty. Näin ollen funktion sisällä määritelty funktio näkyy vain funktion määrittelmässä ohjelmalohkossa. Tällainen funktio on lokaali eli paikallinen. Fuktioiden ulkopuolella pääohjelmassa määritelty muuttuja on globaali. Se ei näy funktion sisällä ellei sitä erikseen määritellä näkyväksi asianomaisen funktion sisällä määrittelyllä:

```
 global $muuttuja;
```

Staattinen muuttuja on funktion sisällä määritelty lokaali muuttuja, joka säilyttää arvonsa koko ohjelman suorituksen ajan funktion kutsukerrasta toiseen. Staattinen muuttuja määritellään static -avainsanalla ensimmäisen kutsukerran yhteydessä ja sille annetaan samalla alkuarvo:

```
static $muuttuja=0;
```

Esimerkki:Mieti, mitä seuraava ohjelma tulostaa ja mitä eroa laskureilla on.

```php
<?php
  function huonoLaskuri()
  {
    $laskuri++;
   print("Huono laskuri: $laskuri <br>");
  }
   function hyvaLaskuri()
   {
    static $laskuri=0;
    $laskuri++;
    print("Hyvä laskuri: $laskuri <br>");
  }

  //Kutsutaan molempia funktioita 4 kertaa
  for($i=1;$i<=4;$i++)
  {
    huonoLaskuri();
    hyvaLaskuri();
  }
?>
```

10 Ympäristömuuttujat

Palvelimen ja selaimen välisessä kommunikoinnissa tallennetaan aina tietoa etukäteen määriteltyihin muuttujiin, ns. ympäristömuuttujiin. Näitä muuttujia kutsutaan usein myös **CGI-muuttujiksi**, koska ne ovat saatavilla lähes aina palvelinympäristössä riippumatta käytetystä ohjelmointiympäristöstä. CGI (Common Gateway Interface)on standardi rajapinta palvelimen ja selaimen välillä. CGI-rajapintaa hyödyntäviä ohjelmia voidaan toteuttaa hyvin monella eri ohjelmointikielellä.

Osa muuttujista on Apachen luomia osa PHP-ympäristön omia funktioita.Täyden listan etukäteen määritellyistä muuttujista saa PHP:ssa funktiolla phpinfo().

Esimerkkejä ympäristömuuttujista:

$CONTENT_TYPE sisältää palvelinkutsussa käytetyn MIME-tyypin, jos on käytetty POST-metodia.

$HTTP_REFERER sisältää edellisen URL-osoitteen, joka viittasi nykyiseen sivuun. Kaikki selaimet eivät lähetä tätä tietoa.

$HTTP_USER_AGENT sisältää lukuohjelman (selain) tyypin.

$REMOTE_ADDR sisältää asiakkaan IP-osoitteen

palvelimen asetuksista riippuen ympäristömuuttujiin voidaan viitata suoraan niiden nimellä tai getenv-funktiota käyttäen. Students.osao.fi-palvelimella on käytettävä pitempää muotoa.

Esimerkki:

```php
<?php
  print "IP-osoitteesi on " .getenv("REMOTE_ADDR") ."<br>";
  print "Käyttämäsi selain on" .getenv("HTTP_USER_AGENT");
?>
```