

Satakunnan ammattikorkeakoulu

Jari-Pekka Multisilta

WWW-OHJELMOINTI PHP-KIELELLÄ

Tietotekniikan koulutusohjelma

Tietoliikennetekniikka

2007

WWW-OHJELMOINTI PHP-KIELELLÄ

Multisilta Jari-Pekka
Satakunnan ammattikorkeakoulu
Tekniikan koulutusohjelma
Tietoliikennetekniikka
Joulukuu 2007
Karri Kivi
UDK:
Sivumäärä:

Avainsanat: Php, MySQL, SQL, Web-ohjelmointi

Opinnäytetyön tarkoituksena oli esitellä www-ohjelmointia mahdollisimman laajasti. Tietokannoista tuli selvittää perusteet, sekä kaikki oleellinen, mikä liittyy www-ohjelmointiin. Php-ohjelmoinnissa keskityttiin siihen, miten saadaan ohjelmoitua toimiva, helppokäyttöinen ja nopea sovellus verkkokaupan käyttöön. Osana opinnäytetyötä suunniteltiin ja toteutettiin www-sovellus Php-kelellä.

Tavoitteisiin päästiin osin testaamalla Php- ja MySQL-tekniikoiden toimivuutta ja osin soveltamalla ja kokeilemalla etsittyä tietoa käytäntöön. Lopputuloksena saatiin koottua varsin laaja kokonaisuus www-ohjelmointitietoa, jonka pohjalta oli helppo lähteä ohjelmoimaan lopullista sovellusta.

WWW-PROGRAMMING IN PHP-LANGUAGE

Multisilta Jari-Pekka

Satakunta University of Applied Sciences

www-programming in Php-language by using MySQL

Information Technology

December 2007

Karri Kivi

UDK:

Number of pages:

Key words: Php, MySQL, SQL, Web-Programming

The meaning of studies was to perform www-programming as widely as possible. We had to sort out basic information about data base and everything relevant that connects with www-programming. In Php-programming we had to focus on how we can program working, practical and fast application for the use of netcommerce. Part of the studies we had to plan and carry through the www-application in Php-language. We reached the goals partly by trying the searched information in real life. As result we managed to collect wide entirety of www-programming information.

SISÄLLYS

1 JOHDANTO	6
2 WWW-OHJELMOINNIN TEKNIIKAT	6
2.1 Ohjelmointikielen valinta.....	7
3 TIETOKANNAT	8
3.1 Yleistä	8
3.2 Käsiteanalyysi	8
3.3 SQL-kieli.....	9
3.3.1 Taulun luonti tietokantaan.....	9
3.3.2 Rivin lisäys tauluun.....	10
3.3.3 Rivien päivitys tauluun	10
3.3.4 Rivin poisto taulusta.....	11
3.3.5 Tiedon haku tietokannasta.....	11
4 PHP-OHJELMOINTI	12
4.1 Yleistä	12
4.2 Syntaksi	12
4.3 Muuttujat.....	14
4.4 Tyypit	14
4.5 Tyypimuunnokset Php:ssä	18
4.6 Operaattorit	19
4.6.1 Aritmeettiset operaattorit	20
4.6.2 Sijoitusoperaattorit	21
4.6.3 Loogiset operaattorit	22
4.6.4 Bittioperaattorit	23
4.6.5 Vertailuoperaattorit	23
4.7 Kontrollirakenteet	24
4.7.1 Toistolauseet	24
4.7.2 Ehtolause	27
4.7.3 Tapauslause	28
4.8 Funktiot	29
5 WEB-SOVELLUKSEN RAKENTAMINEN.....	30
5.1 HTTP-protokolla.....	30

5.2 Php Web-ohjelmoinnissa	31
5.3 Tiedon välitys sovelluksen sivujen välillä	32
5.3.1 Get ja Post	32
5.3.2 Istunnot.....	32
5.3.3 Evästeet	33
5.3.4 Yhteenveto	33
5.4 Tärkeimmät MySQL-funktiot	34
5.4.1 Yhteyden muodostus ja Tietokannan valinta	34
5.4.2 SQL-toiminnot Php:ssa	34
5.4.4 Tietokannan sulkeminen ja haun tuloksen tyhjentäminen	35
5.5 Tiedostojärjestelmä-funktiot	36
5.5.1 Copy	36
5.5.2 Unlink.....	36
5.6 Selainohjelman UPLOAD-toiminto.....	37
6 PHP:N KÄYTTÖ PROJEKTISSA	37
7 YHTEENVETO	40
LÄHDELUETTELO	42

1 JOHDANTO

Tarkoituksena oli luoda sivusto, jossa myydään moottoripyöriä ja niiden osia. Tässä työssä esitellään teoriaa ja tekniikoita, jotka tulivat käyttöön lopullisessa sovelluksessa. Tarkastellaan hieman tietokannan toimintaa ja sitä, miten Php:lla voidaan hallita tietokantoja. Työssä käydään myös läpi kaikki sellaiset funktiot, joita tarvitaan toteuttaessa sovellusta. Sql-kielestä käsitellään perusteet ja yleisellä tasolla relaatiotietokantojen filosofiaa.

Php-kielestä pyrin kertomaan mahdollisimman syvällisesti juuri niistä asioista, joita käytimme työssämme. Yhdessä S. Kanerviston opinnäytetyön kanssa tämä opinnäytetyö on kattava kokonaisuus Php-ohjelmoinnista Apache-palvelinympäristössä. Kanerviston opinnäytetyössä on keskitytty enemmän ohjelmointiympäristön ja lopullisen sovelluksen esittelemiseen, kun taas tässä työssä on kerrottu enemmän ohjelmointitekniikoista ja käytettyjen ratkaisujen toiminnasta.

Tästä työstä voi kuka tahansa etsiä ne tärkeimmät asiat, joita www-ohjelmointi Php:lla sisältää. Tärkeimmässä osassa ovat MySQL-funktiot, lomakkeiden käsittelyfunktiot ja istunnon toteuttaminen.

2 WWW-OHJELMOINNIN TEKNIIKAT

Perinteisesti HTML-kieli perustuu tietystä joukosta standardinimisiä elementtejä eli tageja, jotka kuvaavat dokumentin rakenteen. Näillä voidaan tuottaa staattisia HTML-dokumentteja, joiden päivittäminen ja korjaaminen on hyvin hankalaa ja aikaa vievää. Koska HTML-dokumenttien laatijat halusivat helpomman tavan tuottaa dokumentteja, jotka sisältävät muuttuvaa tietoa, oli keksittävä tapa toteuttaa tämä. (Setola, 2004, 9)

Webin alkuperäistä staattista luonnetta on vuosien kuluessa terästetty useilla erilaisilla tekniikoilla. Tekniikat voidaan jakaa kolmeen ryhmään. (Rantala, 2002, 8)

Web-selaimessa suoritettaviin asiakastekniikoihin

Web-palvelimessa suoritettaviin palvelintekniikoihin

HTTP-yhteyskäytäntöön liittyviin tekniikoihin

Näistä meitä kiinnostavin alue on palvelintekniikat. Palvelintekniikoita ovat

CGI (Käytettyjä kieliä mm. Perl, C, Php.)

Java Servletit

Upotetut tekniikat

- SSI (Server Side Includes)
 - Php (Php: Hypertext Preprocessor)
- Asp (Active Server Pages)
 - Jsp (Java Server Pages)

Sovelluspalvelimet

SSI-tekniikan avulla voidaan dokumentteihin upottaa yksinkertaisia palvelinpään toiminnallisuuksia. SSI mahdollistaa esimerkiksi useilla sivuilla toistuvan navigaatiopalkin upottamisen useaan dokumenttiin. CGI on yleinen liityntä Web-palvelimen ja sen suorittamien ohjelmien välillä. CGI ei ole ohjelmointikieliriippuvainen, vaan CGI:tä voidaan hyödyntää useilla eri kielillä. ASP on lähinnä Microsoftin Web-palvelimien käyttämä upotustekniikka. Ohjelmointikielenä käytetään yleisesti VBScript-kieltä. JSP on Sunin kehittämä Asp:n ja Php:n kaltainen tekniikka. Dokumentti käännetään Java Servletiksi ennen suoritusta. Java Servletit ovat tekniikka, jolla Web-dokumentteja tehdään Java-sovelmien avulla. Servletissä ei voida koodia sekoittaa HTML:n sekaan, vaan kaikki HTML-koodi on generoitava Java-koodilla. (Rantala, 2002,8-10).

2.1 Ohjelmointikielen valinta

Php on HTML-koodin sekaan sijoitettava skriptauskieli, jolla on helppoa saada dynamiikkaa Web-sivulle. Tietokantojen kanssa toimittaessa Php on ehdottoman

helppokäyttöinen ja siksi soveltuu tähän projektiin erittäin hyvin. Lisäksi Php on ilmainen ja syntaksiltaan muistuttaa enemmän C-kieltä kuin esimerkiksi ASP.

3 TIETOKANNAT

3.1 Yleistä

Tietokantojen tehtävä on kerätä ja säilyttää tietoa. Tietokannan tietosäiliö on taulu, johon kaikki tieto talletetaan. Tietokannassa voi olla useita tauluja ja niiden välillä voi olla erilaisia suhteita. Taulussa on rivejä ja jokaisen rivin pitää olla erilainen. Koska jokainen rivi on erilainen, ne voidaan yksiselitteisesti tunnistaa. Tunnistukseen tarvittavaa saraketta tai sarakkeita kutsutaan avaimeksi.

Toisin sanoen taulu näyttää jokseenkin samanlaiselta, kuin excel-työkirja. Eroavaisuutena on, että tietokannasta voidaan hakea yksi tai useampi rivi nopeasti ja näin ollen tietokannan ylläpitäminen on huomattavasti nopeampaa ja tehokkaampaa.

Tietokannan tyypit ovat merkkijonot, numeeriset tyypit, bittitietotyypit sekä aikatietotyypit. Kaupallisissa tietokantaohjelmistoissa ovat mukana myös kuvan ja äänen tallennukseen vaadittavat tallennusmuodot. Numeeriset tyypit voidaan jakaa kokonais- ja liukulukuihin. Merkkijonot ovat joko kiinteä- tai vaihtuvamittaisia merkkijonoja. Aikatieotyyypeistä tärkeimpiä ovat päivä, aika, ajanjakso ja aikaleima.

3.2 Käsiteanalyysi

Käsiteanalyysi on laaja ja tärkeä, jopa tärkein työ tietokantoja luotaessa ja sen rooli korostuu mitä suurempia kantoja ollaan luomassa. Käsiteanalyysissä suunniteltiin millaisia tauluja tarvittaisiin ja mitä tauluihin tulitaisiin tallentamaan. Lopullisesti taulujen sisältö päätettiin, kun tiedettiin miltä projekti kokonaisuudessaan tulisi näyttämään valmistuessaan. Käsiteanalyysistä on hyvä tietää muutamia perusasioita, joita varmasti tulee jossain projektin vaiheessa vastaan.

Kun tietokantaa aletaan suunnittelemaan, on syytä hieman miettiä millainen kanta olisi järkevää tehdä ja mitä sinne kannattaisi tallentaa. Tätä kutsutaan käsiteanalyysiksi.

Käsiteanalyysin tärkeimpiä asioita ovat entiteetit, attribuutit, ja yhteydet. Entiteetit ovat jotain sellaista, mitä halutaan tallentaa tietokantaan. Tässä projektissa halutaan tallentaa moottoripyöriä ja niiden osia. Seuraavaksi pitää ottaa selvää mitä attribuutteja eli ominaisuuksia niistä pitää tallentaa (Merkki, malli ym.). Mikäli entiteeteillä on riippuvaisuuksia toisiinsa, kutsutaan niitä yhteyksiksi. Tässä tapauksessa tällaisia riippuvuuksia ei ole, mutta heti jos esimerkiksi pyörien omistajia olisi useampia (entiteetit: omistaja ja pyörät) olisi niiden välillä riippuvuus, siitä kuka omistaa minkäkin pyörän.

3.3 SQL-kieli

Sql on standardoitu relaatiotietokantojen kyselykieli, jolla voidaan hallita kaikkia tietokannan toimintoja. Sql-kielellä tapahtuu niin lisäys-, poisto- kuin muokkaus-toiminnotkin. Php tarjoaa suoran rajapinnan useimpiin relaatiotietokantoihin, niin myös MySQL-tietokantaan.

3.3.1 Taulun luonti tietokantaan

Ennen kuin mitään voidaan tietokantaan tallentaa on luotava vähintään yksi taulu. Tietokannan taulua luotaessa pitää tietää varmasti sellaiset asiat kuin miten taulua tullaan käyttämään, mihin tarkoitukseen taulu tehdään ja ketkä sitä käyttävät ja millä oikeuksilla. Tietokannan taulun käyttöoikeus voidaan toteuttaa monella tapaa. Yksi mahdollisuus on luoda yhteys toiseen tauluun, jossa on käyttäjät ja salasana. Näin saadaan taulun jokaiselle riville oma käyttäjä ja vain tämä käyttäjä voi riviä lukea.

Tietokantaan lisätään taulu komennolla `CREATE TABLE taulun_nimi ()`. Sulkuihin tulee taulun kentät ja niiden tyypit. Esim.

```
CREATE TABLE mopot (id integer auto_increment primary key,  
Merkki varchar(25),
```

Malli varchar(25),
Kuvaus text)

Ensimmäisessä sarakkeessa oleva auto_increment- määrittely tarkoittaa, että tietokantaohjelmisto antaa automaattisesti sarakkeen arvoksi yhden suuremman kuin edelliselle riville on annettu. Näin saadaan jokaiselle riville yksiselitteinen arvo. Primary key-asetus annetaan juuri tästä syystä tälle sarakkeelle. Tämä id-sarake mahdollistaa sen, että jokainen taulussa oleva rivi on erilainen ja ne voidaan tämän id:n perusteella hakea yksitellen myöhempää käyttöä varten.

3.3.2 Rivin lisäys tauluun

Rivin lisäys tauluun tehdään INSERT-komennolla. Kun edellä tehtyyn taulun halutaan lisätä rivi se voitaisiin tehdä vaikkapa seuraavasti.

```
INSERT INTO mopot (Merkki,Malli,Kuvaus) VALUES ('Su-  
zuki','PV','ihan vaan mopo')
```

Sarakkeiden nimet ja VALUES voidaan myös jättää pois, jos pidetään huoli, että arvot menevät oikeassa järjestyksessä tauluun.

3.3.3 Rivien päivitys tauluun

Mikäli halutaan muuttaa jo tallennetun rivin sisältöä, on se tehtävä UPDATE-komennolla. Jos halutaan muuttaa äsken tallennetun rivin kuvaus-sarakkeen tekstiä, tehtäisiin se näin:

```
UPDATE mopot SET Kuvaus='hieno PV' WHERE id=1
```

Komennossa on oletettu, että tämän rivin id on 1. Ilman where-ehtoa komento muuttaisi kaikkien taulussa olevien rivien Kuvaus sarakkeet arvoon "hieno PV".

3.3.4 Rivin poisto taulusta

Rivi voidaan poistaa taulusta DELETE-komennolla. Delete-komennossa on syytä pitää mielessä, että yksi komento voi tuhota yhden rivin, monta riviä tai vaikka kaikki rivit. Delete-komento tuhoaa aina kaikki rivit, joihin sen määrietykset täsmäävät.

```
DELETE FROM mopot WHERE Merkki='Suzuki'
```

Tämä komento tuhoaa kaikki rivit joissa mopon merkki on Suzuki. Jos halutaan varmistaa, ettei taulusta lähde kuin yksi rivi, where-ehdoksi on aina laitettava rivin pääavain. Tässä tapauksessa se olisi id-sarake.

3.3.5 Tiedon haku tietokannasta

Kaikki tieto haetaan tietokannasta SELECT-lauseella. Select on toimintona hyvin monipuolinen ja lauseesta saa hyvinkin monimutkaisen niin halutessaan. Toisaalta yksinkertaiset haut on helppoja tehdä, mikäli ei käytä mitään erikoishakuja. Aivan perushakuna voidaan pitää sellaista hakua, jossa taulusta haetaan kaikki tieto.

```
SELECT * FROM mopot
```

Mikäli haluttaisiin hakea vain jotain tiettyä saraketta kirjoitettaisiin sen sarakkeen nimi tähden tilalle

```
SELECT Kuvaus FROM mopot
```

Kuten muuallakin SQL:ssä voidaan select-lauseessakin antaa kriteereitä, joiden perusteella haku tehdään. Esimerkiksi voitaisiin tehdä haku, jolla haettaisiin kaikki mopot, joiden id on 50.

```
SELECT * FROM mopot WHERE id=50
```

SQL:ssä voidaan myös ryhmitellä rivejä, jolloin kaikki rivit samalla sisällöllä näkyvät yhtenä rivinä.

```
SELECT DISTINCT Merkki FROM mopot
```

Tässä tulostettaisiin kaikki merkit joita taulusta löytyy. Jos vielä haluttaisiin laittaa merkit aakkosjärjestykseen käytettäisiin ORDER BY -lisämäärettä.

```
SELECT DISTINCT Merkki FROM mopot ORDER BY Merkki
```

4 PHP-OHJELMOINTI

4.1 Yleistä

Php-kielen kehityksen katsotaan alkaneen 1994, kun Rasmus Lerdorf julkaisi muutamia Perl-skriptejä. Nämä skriptit julkaistiin Personal Home Page Tools -pakettina. Myöhemmin php:hen lisättiin Form Interpreter -paketti, joka takasi jo kohtuullisen suosion Web-ohjelmoijien keskuudessa. Varsinainen läpimurto oli Php3:n julkaiseminen. Php3 sisälsi mm. MySQL- ja Apachetuen. Php4 ja Php5-versioissa on tästä vielä parannettu mm. Xml-tukea ja lisätty olio-ohjelmointiominaisuudet. Php on tällä hetkellä suosituin tapa tuottaa dynaamisia verkkosovelluksia.

4.2 Syntaksi

Php on skriptikieli, jota laitetaan sellaisenaan html-koodin sekaan. Php kirjoitetaan html:ssä `<?php` ja `?>`-merkkien väliin. Toimivia ovat myös `<?` ja `?>`-merkit. Php:n syntaksi on hyvin samankaltaista kuin C:n ja Javan syntaksi. Komennot ovat Php:ssa kirjainkoosta riippumattomia, kuten funktiotkin. Merkkäa() funkiokutsu toimisi myös näin merkkAA(). Muuttujissa kirjainkoolla sen sijaan on merkitystä. \$MUUTTUA ja \$muuttuja ovat siis kaksi eri muuttujaa.

Www-ohjelmointia ajatellen Php:ssa on yksi hyvin tärkeä ominaisuus. Koodi voidaan katkaista, väliin kirjoittaa tavallista HTML-koodia ja taas jatkaa Php-koodilla. Esimerkiksi ehtolauseissa ja toistolauseissa tästä on huomattavaa hyötyä.

Kaikkea HTML-koodia ei tarvitse generoida Php:ssa, vaan voidaan kirjoittaa sellaisenaan Php-koodien väliin. Esim.

```
<?
for ($i = 0 ; $i < 10; $i++) {
echo "Terve<br>";
}
?>
```

Tämä siis voidaan tehdä seuraavastikin:

```
<?
for ($i = 0 ; $i < 10; $i++) {
?>

Terve<br>

<? } ?>
```

Vaikka tässä jälkimmäinen vaihtoehto saattaa näyttää jopa hieman kömpelöltä, on sillä ehdottomat hyvät puolensa hiemankaan pidemmässä ohjelmakoodissa.

Kommentit voidaan php:ssa kirjoittaa kolmella eri tavalla. Voidaan kirjoittaa monirivisesti, tai yksirivisesti. Yksirivisiä esitystapoja on kaksi erilaista.

```
<?php
/* monirivinen kommentti näiden
merkkien sisään */
// yksirivinen kommentti
# yksirivinen kommentti toisella tapaa
?>
```

4.3 Muuttujat

Toisin kuin yleisimmissä ohjelmointikielissä, php:ssa ei tarvitse muuttujaa esitellä ennen käyttöä, vaan muuttujaa voidaan suoraan käyttää koodissa. Tästä seuraa tiettyjä ongelmia. Ohjelmakoodin muodostus kyllä nopeutuu jonkin verran, mutta varsinkin pidemmissä ohjelmapätkissä saman muuttujan oikein kirjoituksessa on oltava tarkkana. Php tai selain ei mitenkään reagoi ”kirjoitusvirheisiin”, vaan pitää niitä eri nimisinä muuttujina.

4.4 Tyypit

Boolean

Boolean-tyyppinen muuttuja voi saada arvon True tai False. Boolean tyyppillä voidaan tehdä vertailuja normaaliin tapaan ja hieman yksinkertaisempaan ja fiksumpaan tapaan esim.

```
if ($Muuttuja == TRUE) {  
    echo "on se totta";  
}
```

```
if ($Muuttuja) {  
    echo "on se totta";  
}
```

Molemmissa tavoissa toiminta on sama.

Integer

Integer on kokonaisluku, joka ei siis voi saada desimaaliarvoa. Integer:llä voidaan esittää kymmen-, oktaali- ja hexadesimaalijärjestelmän lukuja.

```
$a = 1234; // positiivinen kokonaisluku  
$a = -123; // negatiivinen kokonaisluku
```

```
$a = 0123; // oktaaliluku
$a = 0x1A; // heksadesimaaliluku
```

Float

Liukuluku on tyyppi, jolla voidaan esittää desimaalilukuja hyvinkin tarkasti. Liukuluvun syntaksia on kolmenlaista, joista varmasti ensimmäinen on käytetyin.

```
$a = 1.234;
$b = 1.2e3;
$c = 7E-10;
```

Liukuluku käyttää 64-bittistä IEEE-standardia.

String

Merkkijono on varmasti käytetyin tietotyyppi ja ehkä se on myös hankalin kaikista tietotyypeistä. Merkkijonon käsittelyyn on monia valmiiksi rakennettuja funktioita, joista enemmän myöhemmin. Merkkijonoon kirjoitettaessa voidaan käyttää kahdenlaisia heittomerkkejä ”-merkkejä ja ’-merkkejä. Niiden toiminnassa on seuraava ero:

```
<?php
$sana="moi";
$merkkijono="moi $sana";
$merkkijono2 ='moi $sana';
echo $merkkijono;
echo $merkkijono2;
?>
```

Tässä kaksi echo-riviä tulostaa:

```
moi moi
moi $sana
```

Toisin sanoen ”-merkkien sisällä voidaan tulostaa myös muuttuja sisältö, mutta ’-merkeissä tulostuu juuri se, mitä niiden välissä on.

Lainausmerkkien sisällä voidaan käyttää myös seuraavia erikoismerkkejä, jotka ovat merkittyjä kenoviivalla:

<code>\n</code>	Rivinvaihto (LF, ASCII 10)
<code>\r</code>	Telanpalautin (CR, ASCII 13)
<code>\t</code>	Tabulaattori (HT, ASCII 9)
<code>\\</code>	Kenoviiva \
<code>\\$</code>	Dollarimerkki \$
<code>\"</code>	Lainausmerkki "

Lisäksi kenoviivalla voidaan kirjoittaa mikä tahansa merkki oktaali- tai heksadesimaalimuodoissa.

```
\123
```

```
\x32
```

Kaksi tai useampia merkkijonoja voidaan yhdistää yhdeksi merkkijonoksi pisteoperaattorilla esim:

```
$teksti = "mopo";
$merkkijono1 = "kesä" . $teksti;
echo $merkkijono1;
```

Tulostus näyttäisi siis tältä:

```
kesämopo
```

Taulukko

Taulukko on muuttuja, joka sisältää useita alkioita. Näihin alkioihin voidaan viitata taulukkomuuttujan indeksillä. Indeksi voi olla luku tai merkkijono. Taulukko luodaan kuten muutkin muuttujat, lisäämällä siihen arvoja esim.

```
$taulukko[1]="tekstiä";
$taulukko[2]="lukuja";
$taulukko[3]=12;
$taulukko[4]=15;
$taulukko["jotain_ihan_muuta"]="ei mitään";
```

Mikäli taulukon indeksinä on merkkijono, on siihen viitattava heittomerkkien sisällä. Pois lukien tilanteet, joissa taulukkoon viitataan merkkijonon sisällä.

```
Normaali käytäntö:
$muuttuja = $taulukko["eka"];

merkkijonon sisäinen käytäntö:
echo "sellaista se elämä on $taulukko[eka]";
```

Jälkimmäisessä tapauksessa siis ei heittomerkkejä tarvita hakasulkujen sisään.

Taulukko voidaan luoda myös ilman indeksinumeroitua, jolloin ensimmäinen indeksi on oletuksena nolla seuraava yksi jne.

```
$taulukko = array("yksi", "kaksi", "kolme");
```

Tai antaen itse indeksit.

```
$taulukko = array("eka" => "ykkönen", "toka" => "kakkonen", "kolmas"
=> "kolmonen");
```

Nämä taulukot käyttäytyvät kuin ne olisi annettu seuraavasti.

```
$taulukko[0]="yksi";
```

```

$taulukko[1]="kaksi";
$taulukko[2]="kolme";

$taulukko["eka"]="ykkönen";
$taulukko["toka"]="kakkonen";
$taulukko["kolmas"]="kolmonen";

```

4.5 Tyypimuunnokset Php:ssä

Php:ssä ei muuttujia esitellä joksikin tietyn tyypiksi, vaan tyyppi mukautuu sisällön mukaan. Mikäli merkkijonolla yritetään suorittaa laskutoimitusta, onnistuu se mikäli merkkijonossa on vain numeroita esim.

```

$merkkijono = "100";
$merkkijono++;

```

Tämän jälkeen tietotyyppi on integer ja arvo on 101. Mikäli merkkijonoa ei voida tulkita miksikään luvuksi saa se arvon nolla.

```

$merkkijono = "jotain tekstiä";
$merkkijono++;

```

Nyt muuttujan tietotyyppi on jälleen integer ja arvo 1.

Boolean-tyyppiseksi tiedoksi käy minkä tyyppinen tieto tahansa. False:ksi tulkitaan seuraavat arvot: false, null, kokonaisluku 0, liukuluku 0.0, tyhjä merkkijono "" ja merkkijono "0", tyhjä taulukko ja olio, jolla ei ole ominaisuuksia. Mikä tahansa muu arvo tulkitaan aina True:ksi boolean-tyypiksi muunnettuna.

```

$muuttuja = "Manu";

if ($Muuttuja) {
    echo "on se totta";
}

```

```
}  
  
if (!$muuttuja) {  
    echo "ei ole totta";  
}
```

Tässäkin tapauksessa muuttujan arvo "manu" tarkoittaa että muuttuja tulkitaan True:ksi.

Hyvin harvoin Php:tä koodattaessa huomaa tarvitsevansa tyyppimuunnoksia, koska se tapahtuu "kuin itsestään". Toki Php:ssäkin on olemassa pakkomuunnin, joka muuttaa tyypin väkisin halutuksi, tällaista kuitenkin tarvitsee tuskin koskaan. esim.

```
$muuttuja = (float) $muuttuja2
```

4.6 Operaattorit

Operaattoreita on monenlaisia ja niille kaikille on omat erityistarkoituksensa. Operaattoreilla on suoritusjärjestyksensä, jonka mukaan suoritus toteutetaan. Suoritusjärjestys ensimmäiseksi suoritettavasta viimeiseksi suoritettavaan:

New
[
++ --
! ~ - (int) (float) (string) (array) (object)
@
* / %
+ - .
<< >>
< <= > >=
=== != ===== !==
&
^
&&
? :
= += -= *= /= .= %= &= = ^= <<= >>=
And
Xor
Or
,

Taulukko

4.6.1 Aritmeettiset operaattorit

Aritmeettiset operaattorit ovat PHP:ssä samankaltaiset kuin muissakin ohjelmointikielissä. Aritmeettisten operaation suoritus voidaan tehdä joko ennen lauseen suoritusta tai sen jälkeen. Mikäli suoritus tehdään jälkeen, se tulee voimaan vasta seuraavalle ohjelmariville siirryttäessä. esim:

```

a=1;
echo "a on: " . a++;           // a on 1
echo a;                         //a on 2
echo ++a;                       //a on 3

```

Aritmeettisiä operaattoreita ovat:

-\$a	Negaatio
\$a + \$b	Yhteen
\$a - \$b	Vähennys
\$a * \$b	Kerto
\$a / \$b	Jako
\$a % \$b	Jakojännös

4.6.2 Sijoitusoperaattorit

Sijoitusoperaattoreista yleisin "=" -operaattori sijoittaa operaattorin perässä olevan tiedon operaattorin edessä olevaan muuttujaan. Operaattoreilla voi myös kikkailulla yhdistelemällä niitä toisiinsa. esim.

```

luku1 = 10;
luku1 -=5;           //luku1 on 5

```

Myös merkkijonoille voidaan tehdä yllä esiteltyjä operaatioita. Esim.

```

$mjono = "kesä ";
$mjono .= "mopo";           //mjono on kesä mopo

```

Operaattoreilla kikkailu voi kyllä pitkällä tähtäimellä olla melkoinen sudenkuoppa, jos ei aina ole varma mitä mikäkin operaattoriyhdistelmä oikeasti tekee.

4.6.3 Loogiset operaattorit

Yleisimmin käytetty on ! eli ei-operaattori, joka on invertteri. Esim:

```
if (!$muuttuja){
...
}
```

tarkoittaa siis sitä, että muuttujan pitää olla epätosi, jotta if lohkoon siirrytään. Ks. Tyypit kappale. Muita loogisia operaattoreita ovat AND, OR ja XOR. Esimerkit kustakin:

```
if ($muuttuja1 AND $muuttuja2) {
...
}
```

```
if ($muuttuja1 OR $muuttuja2) {
...
}
```

```
if ($muuttuja1 XOR $muuttuja2) {
...
}
```

Ensimmäisessä if-lohkoon mennään, jos molemmat ehdot täyttyvät. Toisessa silloin kun edes jompikumpi ehdoista täyttyy ja kolmannessa kun VAIN toinen ehdoista täyttyy. AND ja OR sanat voidaan korvata myös && ja || merkeillä, jotka ovat ehkä hieman tavanomaisempia. Näitäkin operaattoreita voidaan tarvittaessa yhdistellä. Tässäkin tapauksessa pitää olla erittäin varovainen, että toiminta varmasti on halutun laista. esim.

```
if ($muuttuja1 XOR ($muuttuja2 AND $muuttuja3) {
...
}
```

Tässä siirrytään if-lohkoon, kun muuttuja1 tai muuttuja2 ja 3 täyttävät ehdon. Huomaa sulkeet, ilman niitä toiminta olisi aivan erilaista. Kaikki loogiset operaattorit:

$\$a$ and $\$b$	And
$\$a$ or $\$b$	Or
$\$a$ xor $\$b$	Xor
! $\$a$	Not
$\$a$ && $\$b$	And
$\$a$ $\$b$	Or

4.6.4 Bittioperaattorit

Bittioperaattorit ovat tietyissä sovelluksissa erittäin käyttökelpoisia, mutta ihan yksinkertaisissa sovelluksissa niitä tuskin tarvitaan. Bittioperaattoreita ovat:

$\$a$ & $\$b$	And
$\$a$ $\$b$	Or
$\$a$ ^ $\$b$	Xor
~ $\$a$	Not
$\$a$ << $\$b$	Siirto vasemmalle
$\$a$ >> $\$b$	Siirto oikealle

Bittioperaattoreita ei pidä sekoittaa loogisiin operaattoreihin. Bittioperaattorit toimivat aina ”bitti kerrallaan”.

4.6.5 Vertailuoperaattorit

Vertailuoperaattoreita käytetään vertaillessa muuttujia keskenään tai vertaillessa muuttujia johonkin arvoon. Kaikki vertailuoperaattorit:

$\$a == \b	Yhtä suuri kuin
$\$a === \b	Identtinen
$\$a != \b	Erisuuri
$\$a <> \b	Erisuuri
$\$a !== \b	Ei identtinen
$\$a < \b	Vähemmän kuin
$\$a > \b	Enemmän kuin
$\$a <= \b	Vähemmän tai yhtä paljon kuin
$\$a >= \b	Enemmän tai yhtä paljon kuin

Vertailua käytetään ehto- ja toistolauseissa esim

```
if ($muuttuja==23){
...
}
```

4.7 Kontrollirakenteet

Kontrollirakenteet ajavat ohjelman haluttuun toimintaan, halutulla tavalla. Kontrollirakenteita kuvataan tyypillisesti vuokaaviolla, josta nähdään missä tapauksissa ohjelman pitää tehdä mitäkin. Perinteisessä ohjelmoinnissa vuokaavio on usein se lähtökohta, josta ohjelmaa aletaan rakentamaan. Webbimaailmassa vuokaavio ei niin suureen rooliin koskaan pääse, mutta saattaa tietyissä asioissa helpottaa suurestikin tulevaa ohjelmointityötä.

Kontrollirakenteita ovat erilaiset ehto-, toisto- ja valintalauseet.

4.7.1 Toistolauseet

Yksinkertaisimmillaan ohjelman ohjausta suorittavat toistolauseet, joita on erilaisia ja joiden toiminta hieman poikkeaa toisistaan. Yhteistä näille on että ne tiettyjen ehtojen täytyessä toistavat itseään määrätyn (tai määrittämättömän) ajan.

For-silmukka

For on toistolauseista yleisin ja se onkin lähes jokaisessa ohjelmointikielessä syntaksiltaan samankaltainen. For lausetta käytetään toistamaan tiettyä lausetta tai lauseita määrätyn ajan.

For-lauseen rakenne on for (alku; loppu; toiminta), jossa sulkujen ensimmäinen toimenpide on muuttujan alustus seuraavassa kerrotaan mihin asti for-lause pyörittää silmukkaa ja viimeinen mitä muuttujalle tehdään jokaisella kierroksella.

```
for ($laskuri=1; $laskuri<10; $laskuri++) {
...
}
```

Tässä alustetaan laskuri muuttuja ykköseksi ja kerrotaan, että silmukkaa pyöritetään kunnes laskuri on kymmenen Lopuksi laskuria kasvatetaan yhdellä, joten tämä silmukka toistetaan kymmenen kertaa. Mikäli laskuriin halutaan jokin muu askellus voidaan se hoitaa vaikkapa näin:

```
for ($laskuri=1; $laskuri<10; $laskuri= $laskuri+$laskuri) {
...
}
```

Tämä kasvattaisi laskuria seuraavasti:

```
1
2
4
8...
```

Kun for-lauseen on tarkoitus toistaa vain yhtä lausetta, ei aaltosulkuja tarvita. Kuitenkaan puolipistettä ei tässäkään tapauksessa saa for lauseen perään laittaa. Esim.

```
for ($laskuri=1; $laskuri<10; $laskuri= $laskuri+$laskuri)
echo $laskuri;
```

While-silmukka

While-lause on toiminnaltaan samantapainen kuin for-lausekin.

```
$muuttuja=1;
While ($muuttuja<10){
echo $muuttuja;
$muuttuja++;
}
```

Kuten voidaan huomata, while-lauseessa muuttuja pitää alustaa ennen kuin varsinaiseen while-lauseeseen mennään. Myös laskurin kasvatus pitää tehdä omalla rivillään, joten tämä lisää koodin pituutta jonkin verran. Pienessä koodipätkässä tämäkin on melko mitätön juttu, mutta kun ajatellaan sovellusta, jossa erilaisia toistorakenteita joudutaan käyttämään ehkä useita kymmeniä jopa satoja kertoja voidaan for-lauseella kutistaa helposti koodia muutamilla kymmenillä riveillä. Toisaalta while-toistorakenne on monipuolisempi käytössä.

Do while -silmukka

Tämä toistorakenne on aivan kuin while sillä erotuksella, että while-silmukassa ei silmukkaa suoriteta välttämättä kertaakaan. Do while -rakenteessa silmukka joka tapauksessa suoritetaan vähintään kertaalleen. Esim.

```
$muuttuja=1;
do {
echo $muuttuja;
$muuttuja++
```

```
} while ($muuttuja<10);
```

Normaalitapauksessa tulostus on samanlainen while-silmukan kanssa, mutta entä seuraavassa:

```
$muuttuja=10;

While ($muuttuja<10){
echo $muuttuja;
$muuttuja++;
}

do {
echo $muuttuja;
$muuttuja--;
} while ($muuttuja<10);
```

Esimerkki saattaa tuntua keksimällä keksityltä, mutta epähuomiossa tällaisia ”virhetilanteita” voi kyllä syntyä. Tässä siis ensimmäiseen while-silmukkaan ei mennä milloinkaan ja näyttäisi, ettei toinenkaan tee kuin yhden kierroksen, vaan oikeasti kyseessä on ikuinen silmukka. Ensimmäisellä kerralla suoritetaan muuttujan decrementointi ja arvo onkin nyt 9 ja silmukka menee seuraavalle kierrokselle ja jälleen arvo pienenee. Tarkkana pitää siis tässäkin olla, virhettä kun on niin kovin vaikea löytää suuresta määrästä koodia.

4.7.2 Ehtolause

Ehtolause eli if-lause, jota monessa esimerkissä jo onkin käytetty, on ohjausrakenne, joka ohjaa suorituksen if-rakenteen sisään, mikäli tietyt ehdot täyttyvät. Yksinkertaisimmillaan asia on tämän kaltaista:

```
if ($muuttuja==1)
echo $muuttuja;
```

Kun if-lohkon sisään halutaan useampia kuin yksi lause, on siinä käytettävä aaltosulkeita aivan kuten kaikessa muussakin php-ohjelmoinnissa.

Hieman monimutkaisemman if-lauseesta saa kun siihen lisätään useampia lohkoja. Ensimmäinen lohko on aina if, mutta seuraava on tapauksesta riippuen elseif tai else.

```
if ($muuttuja==1) {  
    echo $muuttuja;  
}  
  
elseif ($muuttuja==2) {  
    echo $muuttuja;  
}  
  
else {  
    echo ("muuttujan sisältö ei ollut yksi eikä kaksi")  
}
```

4.7.3 Tapauslause

Switch-case rakenne on ehdottomasti fiksumpi ratkaisu kuin monen elseif-rakenteen peräkkäin asettelu. Molemmilla saadaan sama tulos aikaiseksi, mutta switch-case on huomattavasti selkeämpää koodia. Tällä rakenteella voisi korvata melkeinpä kaikki if-lauseet aivan yksinkertaisinta tapausta lukuun ottamatta.

```
switch ($muuttuja) {  
    case 1:  
        echo ("luku on 1");  
        break;  
    case 2:  
        echo ("luku on 2");  
        break;  
    case 3:
```

```
echo ("luku on 3");
break;
```

```
default:
```

```
echo ("luku ei ollut mitään 1:den ja 3:n väliltä")
```

```
}
```

Jos mikään arvo ei vastaa muuttujan arvoa ohjataan suoritus default-osaan.

4.8 Funktiot

Funktiot eli aliohjelmat ovat pääohjelmasta irrallisia ohjelmapätkiä, jotka tekevät yleensä jonkin pienemmän kokonaisuuden. Aliohjelmia kutsutaan pääohjelmasta tai toisesta aliohjelmasta. Aliohjelma voi palauttaa arvon. Syy, miksi aliohjelmia käytetään, on niiden tapa lyhentää ohjelmakoodia. Kun tiedetään, että jotain ohjelmapätkää tullaan käyttämään useaan kertaan ohjelman aikana, on ehkä viisasta tehdä siitä aliohjelma.

Funktion voi määritellä missä kohdassa koodia tahansa, mutta yleinen käytäntö on, että se määritellään erilliseen tiedostoon, joka sitten varsinaisessa ohjelmatiedostossa otetaan käyttöön yleensä ensimmäisellä rivillä. Toinen yleinen tapa on sijoittaa kaikki funktiot tiedoston loppuun, tämä kuitenkin estää funktioiden käytön muissa ohjelmissa, joten tiedoston loppuun ei voida globaaleita funktioita laittaa.

```
<html>
<?php
Funktion laske_yhteen($luku1 , $luku2) {
    $tulos = $luku1 + $luku2;
    return $tulos;
}
?>
```

```
</html>
```

Tämä koodi voidaan siis kirjoittaa vaikkapa `funktio.php` nimiseen tiedostoon. Funktiolla on kaksi parametria jotka annetaan funtiokutsussa.

Kun halutaan tämä koodi käyttöön pääohjelmassa, pitää se tehdä esimerkiksi `require()`-funktiolla, joka saa parametrikseen sisällytettävän tiedoston nimen.

```
<html>
<?php
require(funktio.php);
echo "seuraavassa lasketaan muutamia yhteenlaskuja";

echo "1+2".laske_yhteen(1, 2);
echo "20+2" .laske_yhteen(20, 2);
echo "4+3" .laske_yhteen(4, 3);

?>
</html>
```

Kuten voidaan huomata samalla koodipätkällä voidaan tehdä laskuja eri luvuilla. Nyt kun `funktio.php` on jo tehty, voidaan sinne mielin määrin lisätä omia funktioita ja ne ovat automaattisesti käytössä pääohjelmassa.

5 WEB-SOVELLUKSEN RAKENTAMINEN

5.1 HTTP-protokolla

HTTP-protokolla koostuu aina kahdesta HTTP-viestistä asiakaspyynnöstä (HTTP request) ja palvelimen vastauksesta (HTTP response). Asiakaspyyntö sisältää pyyntörivin `http`-otsakkeet ja datan. Pyyntörivi koostuu käytettävästä metodista pyydettävästä dokumentista ja `http:n` versiosta. Mahdolliset otsakkeet lähetetään riveittäin yksi otsake per rivi. Otsake koostuu nimestä arvosta ja rivinvaihtomer-

kistä. Otsakkeidentyyppillinen sisältö on selaimen tyyppi haluttu kieli ym. Pyyntö lopetetaan kahteen rivinvaihtomerkkiin. Mikäli erillistä dataa ei lähetetä tämä tehdään heti otsakkeiden jälkeen.

Palvelimen saatua pyynnön asiakkaalta se vastaa siihen. Ensimmäisellä rivillä on HTTP:n versio, tilakoodi ja sanallinen selitys. Seuraavaksi tulevat http-otsakkeet jälleen riveittäin. Tässä palvelin voi kertoa tietoa itsestään, vastauksen luontiajan, dokumentin tyyppin tms. tietoa. Tämän jälkeen palvelin lähettää datan, joka voi olla vaikka Php:llä luotu HTML-dokumentti. Tyypillisesti Request-Response - tapahtuma voisi olla seuraavan kaltainen.

Request:

GET /index.html/ http/1.0

User-Agent: IE6.0

Accept-Language: fi

Response

http/1.1 200 OK

Server: Apache/2.0

Content-Lenght:278

Content-Type: text/html

<TITLE>html dokumentti index.html </TITLE>

...

Tästä selviää periaate, miten HTTP-protokollalla siirretään tietoa. Sovelluksen toiminta muodostuu useista tällaisista Reguest-Response -toiminnoista.

5.2 Php Web-ohjelmoinnissa

Php:llä voidaan tehdä yksinkertaisia lyhyitä ja helppoja pieniä lisäyksiä html-koodin sekaan, mutta php:llä voidaan myös rakentaa oikeita toimivia tietokantapohjaisia www-sovelluksia, joiden toiminta on riippuvaista käyttäjän toimista sivulla. Php:ssä on monia hyviä ohjelmointia yksinkertaistavia asioita, kuten täydellinen MySql-rajapinta.

Selvää on, että kun käytetään jotain ohjelmointikieltä html-koodin generoimiseen, on suunnittelu, ohjelmointi ja testaustyö kova urakka. Toisaalta, kun kerran on hyvä ja toimiva sovellus saatu aikaan, on sen käyttäminen melko vaivatonta.

Projektia edeltävä tilanne oli, että jokaiselle myytävälle moottoripyörälle oli tehtävä oma html-sivunsa. Valmiissa projektissa on vain yksi Php:llä tehty sivu, tälle sivulle vain haetaan sen pyörän tiedot, jota asiakas haluaa juuri sillä hetkellä katsoa.

5.3 Tiedon välitys sovelluksen sivujen välillä

Www-sovelluksissa on aina tarpeen saada siirrettyä tietoa sivulta toiselle, siinäkin tapauksessa, että käytössä on tietokanta. Tämä tieto on yleensä sellaista, mitä tarvitaan, joko sivun sisältöä luodessa tai tietokantaan tallennettaessa jotain.

Tiedonsiirtotapoja on useita. Vanhin tapa on siirtää tieto http:n header-osiossa GET- tai POST-metodilla. Nykyaikaisempi tapa on käyttää php:ssakin olevaa session-tekniikkaa. Samankaltainen tapa on myös evästeet, joihin voidaan tallentaa samankaltaisesti tietoa kuin sessionmuuttujiinkin.

5.3.1 Get ja Post

Yksinkertaisin ja vanhin tapa siirtää tietoa sivulta toiselle, ovat Get- ja Post-metodit. Nämä kaksi tapaa ovat http-protokollan mukanaan tuomia toimintoja, joissa tieto saadaan siirrettyä http:n otsakkeissa. Get on selväkielinen ja näkyy selaimen osoiterivillä, kun taas Post piilottaa tiedot käyttäjältä näkymättömiin.

Tämän tavan ongelmana on sen vaatima tiedon siirto jokaisen sivun välillä. Esim. istunnoissa ohjelmoijan ei jokaisella sivulla tarvitse antaa muuttujalle arvoa, vaan arvo säilyy koko istunnon ajan.

5.3.2 Istunnot

Session on nykyaikaisempi tapa siirtää tiettyjä asioita sivulta toiselle. Tämän ero edelliseen tapaan on se, että session-muuttuja säilyy koko sen ajan kun ollaan

istuntoon kuuluvassa ohjelmassa. Tämä ei kuitenkaan tarkoita sitä, etteikö Get- ja Post-metodeita tarvittaisi. Joitain asioita ei voi session tekniikalla tehdä ja niissä on yhä käytettävä näitä vanhempia tekniikoita. Käyttäjien hallinnassa ym. asioissa session on kyllä korvaamaton tekniikka.

Istunnot käynnistetään `session_start()`-funktiolla. Tämä on oltava jokaisessa istuntoon kuuluvan ohjelman alussa. Istunto lopetetaan joko sulkemalla selain tai funktiokutsulla `session_unset()`, joka tyhjentää kaikki session-muuttujat. Session-muuttuja on vaikkapa tämän näköinen: `$_SESSION['muuttuja'] = 10;`. Tässä asetettiin session-muuttujan arvoksi 10.

5.3.3 Evästeet

Evästeillä voidaan myös rakentaa samanlaisia toimintoja kuin istunnoillakin (itse asiassa evästeillä voidaan toteuttaa myös istunto). Evästeet eivät välttämättä tarvitse istunnon käynnistämistä toimiakseen, mutta eivät ne sitä mahdollisuutta poiskaan sulje. Evästeen toiminta perustuu siihen, että selain tallentaa muistiinsa halutun muuttujan siihen saakka kun se käsketään sieltä poistaa. Keksi asetetaan `setcookie("Evästeen_nimi", "$_Evästeen_sisältö", 0, "/")` funktiolla. Evästeen asettamisessa kolmas parametri tarkoittaa voimassaoloaikaa päivissä ja neljäs parametri, mihin hakemistoon eväste vaikuttaa (kautta-viiva tarkoittaa, että eväste on voimassa koko domainin vaikutusalueella).

Evästeen ehdottomasti tärkein sovellusalue on tilatietojen ylläpitäminen. Kun tilatieto tallennetaan selaimen muistiin, käyttäjä voi käydä jollain muulla sivustolla ja palata takaisin palveluun minkään tiedon välillä katoamatta.

5.3.4 Yhteenveto

Dynaamisella sovelluksella on aina tarpeen siirtää tietoa sivulta toiselle. Tähän on muutamia ratkaisuja, jotka kaikki ovat aivan käyttökelpoisia. Parhaan sovelluksen toiminnaltaan saa yhdistelemällä näitä tekniikoita toisiinsa. Kaikilla näillä on omat erityispiirteensä ja alueensa, joissa ne ovat lähes korvaamattomia.

5.4 Tärkeimmät MySQL-funktiot

Php:ssa on melko joukko MySQL-funktioita, jotka tekevät mitä erilaisempia asioita. Tietokantojen peruskäytössä kuitenkin tarvitaan pääsääntöisesti vain muutamia näistä. Tässä käydään läpi työn vaatimat tärkeimmät funktioit.

5.4.1 Yhteyden muodostus ja Tietokannan valinta

Yhteyden muodostus MSQL-palvelimeen on hyvin yksinkertaista. Normaalityypisessä tapauksessa tarvitaan vain palvelimen osoite, käyttäjä, nimi ja salasana. Nämä annetaan parametrina `mysql_connect` funktiolle. Esim.

```
$link = mysql_connect('mysql_host', 'mysql_user', 'mysql_password')
```

Tietokanta valitaan antamalla yksinkertaisesti vain tietokannan nimi ja juuri luodun yhteyden resurssinimi parametrina `mysql_select_db`-funktiolle.

```
mysql_select_db('my_database', $link)
```

5.4.2 SQL-toiminnot Php:ssa

Php:ssa voidaan kaikki toiminnot tietokannan kanssa hoitaa yhdellä funktiolla. Funktion käyttö on erittäin helppoa, jos osaa SQL-kielen. Funktiolle annetaan parametrina SQL-lause, jonka sitten palvelin suorittaa. Tästä syystä tietokantaan lisäys sieltä poisto tai haku sujuu yhdellä funktiolla.

```
$query = 'SELECT * FROM my_table'  
$result = mysql_query($query)
```

Tässä `$query`-muuttujaan tallennetaan SQL-lause, joka sitten välitetään `mysql_query`-funktiolle. Saatu tieto tallennetaan `$result`-muuttujaan.

5.4.3 Tiedon muuttaminen HTML-muotoon

Kaikki tieto, mikä saadaan tietokannasta pitää muuttaa HTML-muotoon, jotta selain voi tulkita sen HTML-dokumentiksi. Yksi tapa on käyttää `mysql_fetch_array`-funktiota ja muuttaa tieto HTML-taulukoksi.

```
echo "<table>\n";
while ($line = mysql_fetch_array($result, MYSQL_ASSOC)) {
    echo "\t<tr>\n";
    foreach ($line as $col_value) {
        echo "\t\t<td>$col_value</td>\n";
    }
    echo "\t</tr>\n";
}
echo "</table>\n";
```

Esimerkissä luodaan ensin taulukko, jonka jälkeen pyöritetään silmukkaa niin kauan kuin tietoa löytyy `$result`-muuttujasta. Jokaisessa silmukassa on vielä sisäinen silmukka, jota pyöritetään niin kauan kuin sarakkeita löytyy. Jokaisen sarakkeen sisältö laitetaan `<TD>` ja `</TD>`-elementtien väliin ja jokainen tällainen kokonaisuus `<TR>` ja `</TR>`-elementteihin. Tällä tavoin saadaan muodostettua hyvin yksinkertainen HTML-taulukko tietokannan sisältämästä tiedosta. `MYSQL_ASSOC`-parametri määrittää, että rivin tietoon voidaan viitata sarakkeen nimen perusteella. Muut vaihtoehdot olisivat `MYSQL_NUM` ja `MYSQL_BOTH`, joissa ensimmäisessä voidaan viitata sarakkeen numeroon ja toisessa kumpaan tahansa.

5.4.4 Tietokannan sulkeminen ja haun tuloksen tyhjentäminen

Jotta ohjelma toimisi tehokkaasti, on saatu haku aina syytä tyhjentää. Tyhjentäminen tehdään vapauttamalla muuttuja `mysql_free_result`-funktiolla.

```
mysql_free_result($result);
```

Tietokantayhteys suljetaan `mysql_close`-funktiolla, jolle annetaan parametrina linkki-muuttuja, jota käytettiin avaamaan tietokantayhteys (ks. yhteyden muodostus ja tietokannan valinta).

```
mysql_close($link);
```

5.5 Tiedostojärjestelmä-funktiot

Tiedostojärjestelmä funktioista käytettiin kahta: `Copy` ja `Unlink`. Molempia tarvittiin kuvatiedostojen ylläpitoon.

5.5.1 Copy

`Copy` toimii aivan kuten `Dos:n` tai `Unixin` kopiointi. Funktiolle annetaan alkuperäinen tiedosto ja kohde johon tiedosto kopioidaan. Uusimmissa `Php:n` versioissa molemmat, sekä lähde että kohde voivat olla `URL`ejä.

```
<?php
$file = 'example.txt';
$newfile = 'example.txt.bak';

if (!copy($file, $newfile)) {
    echo "failed to copy $file...\n";
}
?>
```

5.5.2 Unlink

Myös `Unlink` on hyvin yksinkertainen funktio käyttää. `Unlink`:lle annetaan parametrina tiedoston nimi ja hakemisto siinä tapauksessa, jos se on eri kuin `Php`-koodin hakemisto.

```
<?php
$file="data.dat";
if (!unlink($file)){
```

```

    echo "ei voida poistaa tiedostoa";
}
?>

```

5.6 Selainohjelman UPLOAD-toiminto

Uudet selaimet tukevat tiedostojen Upload-toimintoa, jonka avulla voidaan helposti siirtää tiedostot käyttäjän tietokoneelta palvelinkoneelle. Uploadia käytetään kuin mitä tahansa html:n input-elementtiä. Tästä syystä voidaan tiedostojen siirto ja muu tarvittava informaatio siirtää samalla lomakkeella palvelinkoneen lomakkeenkäsittelyohjelmalle, joka tekee tarvittavat toimenpiteet tiedolle. Huomattava on, ettei Upload vielä tallenna tiedostoa pysyvästi palvelimelle, vaan se tallentuu johonkin väliaikaiseen paikkaan muistiin tai kovalevyn sivutusalueelle. Tästä syystä on erikseen vielä lomakkeen käsittelijässä tehtävä tiedoston tallennus haluttuun paikkaan halutulle nimelle.

6 PHP:N KÄYTTÖ PROJEKTISSA

Tässä kappaleessa esitellään muutamia huomionarvoisia koodi-pätkiä. Osa näistä pätkistä käytettiin lähes jokaisella sivulla, osaa vain kerran koko sovelluksessa. Osa koodista on lyhennetty tai muuten yksinkertaistettu luettavuuden säilyttämiseksi. Osa koodista on sellaista, ettei se yksinään toimi, vaan vaatii jonkinlaista ohjausta (kuten esimerkiksi taulukon rivien värin vaihtelu).

Moottoripyörät haluttiin listata siten, että rivien väri vaihtelee. Värin vaihtelu tehtiin seuraavalla koodipätkällä.

```

if ($heiluri==1){
    echo "\t<tr bgcolor=ff6600>\n";
    $heiluri++;
}
else {
    echo "\t<tr bgcolor=ff9933>\n";
    $heiluri--;
}

```

```
}
```

Heilurille annetaan ensin arvo yksi. Kun tätä koodia toistetaan useampia kertoja, arvo vaihtelee välillä 1-2. Tällöin joka toinen rivi on hieman eri värinen.

Pyora2-sivulla olevat kuvien linkit tehtiin Url-koodauksella. Jos tietokannassa on kuvan nimi, siitä generoidaan linkki takaisin tälle sivulle. linkillä on tällöin parametrina kuvan numero ja pyörän id.

```
if($row[7]!="Null" )
    printf ('<a href="pyora2.php?id=%s&kuva=1">1</a>
', $id);
```

Tiedosto poistetaan, jos tietokannassa on kuvan nimi. Tiedostojärjestelmässä oleva nimi muodostuu aikaleimasta kuvan numerosta ja kuvan tietokannassa olevasta nimestä.

```
if ($kuva1!="Null"){
    $tiedosto="./data/".$aikaleima"."1"."$kuva1";
    unlink ("$tiedosto");
```

Monessa kohtaa sovellusta käytettiin toiminnan ohjauksessa metodin tarkistusta. Kun metodi on POST tehdään yleensä jotain lisäystä tai poistoa tietokantoihin.

```
if ($_SERVER["REQUEST_METHOD"]=="POST")
```

Isset-funktiolla voidaan tarkistaa, onko jotain muuttujaa asetettu. Tässä tarkistetaan onko kuva4 asetettu, jos näin on kutsutaan tallenna-funktiota.

```
if (isset($kuva4))
{
    $kuvano=4;
    tallenna($kuva4, $kuva4_name, $datadir, $urldir, $aikaleima, $kuvano);
```

```
}
```

Mysql_fetch_array-funktiota käytettiin lähes kaikissa taulukon muodostus tapah-
tumissa. Huomattavaa on värin vaihto joka rivillä. Jokaisesta rivistä muodostetaan
lomake, jolla on hidden-kentässä pyörän id. Tätä id:tä tarvitaan myöhemmin kun
pyöra2-sivua muodostetaan.

```
while ($line = mysql_fetch_array($vastaus, MYSQL_ASSOC)) {
    // MUODOSTETAAN TAULUKKO SAATUJEN TIETOJEN
    PERUSTEELLA
    if ($heiluri==1){
        echo "\t<tr bgcolor=ff6600>\n";
        $heiluri++;
    }
    else {
        echo "\t<tr bgcolor=ff9933>\n";
        $heiluri--;
    }

    echo '<form action="pyora2.php" method="GET">';

    echo '<td>';
    if($line["merkki"]!="Null")
        printf ("%s", $line["merkki"]);
    echo "</td>\n";

    echo "\t\t<td>";
    if($line["malli"]!="Null")
        printf ("%s", $line["malli"]);
    echo "</td>\n";

    echo '<input type="hidden" name="id" value="'. $line["id"]. "'>';
    echo "\t\t<td>";
```

```

echo '<input type="submit" value="Tiedot">';
echo "</td>\n";

echo "</form>";
}

```

Pyora2-sivulla tulostetaan haluttu kuva image-elementtiä käyttäen. Tiedoston nimi generoidaan tässäkin tietokannasta saadusta tiedosta.

```

if ($kuva==1)
    printf ('<image src=".admin\data\%s1%s"
width="640" heihgt="480"> ', $row[17], $row[7]);

```

Aikaleima pitää muodostaa nimenomaan Php:ssa. Php:n pitää tietää tämä aikaleima. Aikaleima luotiin alun perin siitä syystä, että kuvien nimet saatiin pysymään hallinnassa, ja että osien hallinta olisi helpompaa. Aikaleimalla on siis eri tarkoitus osien ja moottoripyörien hallinnassa.

```

$format = '%d-%m-%Y-%H-%M-%S';
$saikaleima = strftime($format);

```

7 YHTEENVETO

Php ja MySQL ovat yhdessä monipuolinen ja tehokas tapa tehdä verkkosovelluksia. Php:n vahvuudeksi voidaan laskea sen sisältämä erittäin hyvä tuki MySQL-tietokannalle. Koska MySQL-tuki on tehty pohjautumaan SQL-kieleen, on sen käyttö helposti hallittavissa kunhan osaa SQL-kielen perusteet. Php:ssä on myös monia muita sisäänrakennettuja toimintoja, joita muissa ohjelmointikielissä ei vielä ole. Php:n syntaksin omaksuminen ei vie juurikaan aikaa, jos hallitsee en-tuudestaan C-kielen syntaksin. Oikeastaan Php:ssä suurin ero on muuttujien tyy-pittömyys.

Tekijät oppivat Php-ohjelmoinnista projektin kuluessa paljon uusia asioita. Opettiin, että aina hankaliin tilanteisiin ei ole helppoa löytää ratkaisua. Yllättävää olikin, ettei Php-ohjelmointiin löydy lainkaan niin helposti informaatiota, kuin voisi kuvitella. Php:sta on kyllä kirjoitettu paljon, mutta todellisia vinkkejä oikeaan www-ohjelmointiin on silti usein etsittävä hyvin kauan. Ehkä paras paikka hakea ohjelmointiapua on `Php_manual_fi`-niminen dokumentti, jonka löytää www.php.net-sivulta.

LÄHDELUETTELO

Heinisuo, R. (2004). Php ja MySQL Tietokantapohjaiset verkkoratkaisut. Jyväskylä: Gummerus Kirjapaino Oy.

Mehdi Achour, Friedhelm Betz, Antony Dovgal, Nuno Lopes, Philip Olson, Georg Richte, Damien Seguy, Jakub Vrana (1997-2005). PHP Manual [verkkodokumentti]. www.php.net/docs.php. PHP Documentation Group.

Mäkeläinen, J-P. (2004). Opinnäytetyö: Projektinhallintajärjestelmä. Pori.

Noutio, T. (2004). Opinnäytetyö: MySQL- ja PHP-tietokantajärjestelmä. Pori.

Rantala, A. (2002). PHP, WEB-ohjelmoinnin peruskirja. Porvoo WS Bookwell.

Setola, T. (2004). Opinnäytetyö: Web-pohjaisen TET –Järjestelmän suunnittelu ja toteutus PHP:llä. Pori.

Zandstra, M. (2001). PHP Trainer Kit. Helsinki: Oy Edita Ab.