swedish networking technologies

Dynamisk webbprogrammering med PHP

Thomas Höjemo, SNT 2006.

Innehållsförteckning

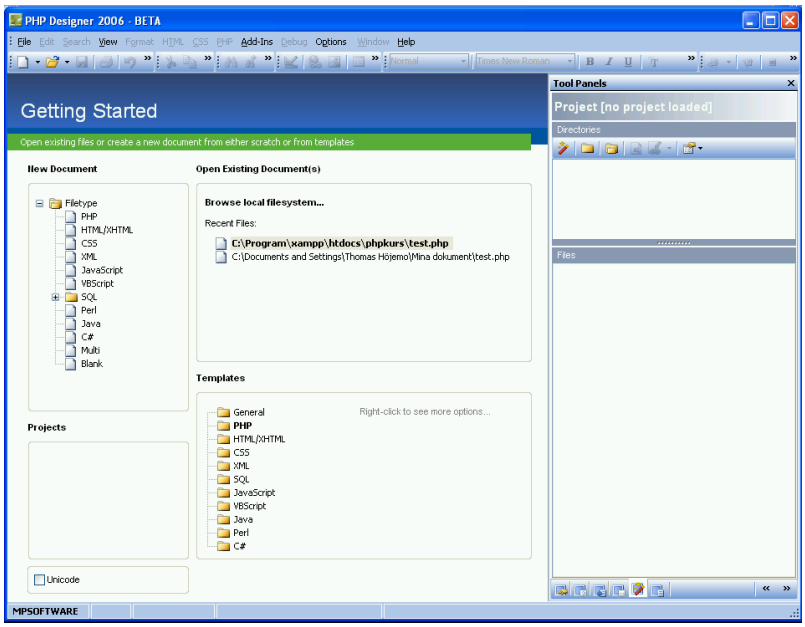
Redigeringsprogrammet PHP Designer

För att programmera i PHP kan man i princip använda en enkel textredigerare som Anteckningar i Windows. Ett mer specialanpassat program ger dock flera fördelar som bland annat färgkodning och snabbtest av koden. För Linux finns det flera bra gratis redigerare som Quanta och Bluefish m.fl. För Windows kostar de flesta programmen pengar. PHP Designer är ett av få Windows-program som finns i en gratisversion.

PHP Designer

PHP Designer är en redigerare som är skräddarsydd för PHP. PHP Designer 2006 går att ladda ned helt gratis från http://www.snapfiles.com/get/mpsphp.html . Den nyaste versionen PHP Designer 2007 som finns på adressen http://www.mpsoftware.dk är dock endast gratis för personligt bruk.

Startfönstret

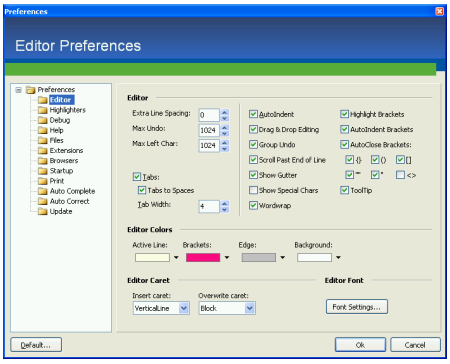


New Document till vänster används för skapa en ny PHP- eller HTML-fil.

Open Existing Documents... i mitten ger snabbåtkomst till senast öppnade filer.

Övriga delar av fönstret är inte relevanta i detta läge.

Inställningar



Genom att gå in på menyn **Options** och undermenyn **Editor preferences...** kan inställningar göras för PHP Designer. Under mappen **Editor** till vänster och **Font Settings**... kan vi byta till typsnittet Courier New i storlek 12 som är lämplig.

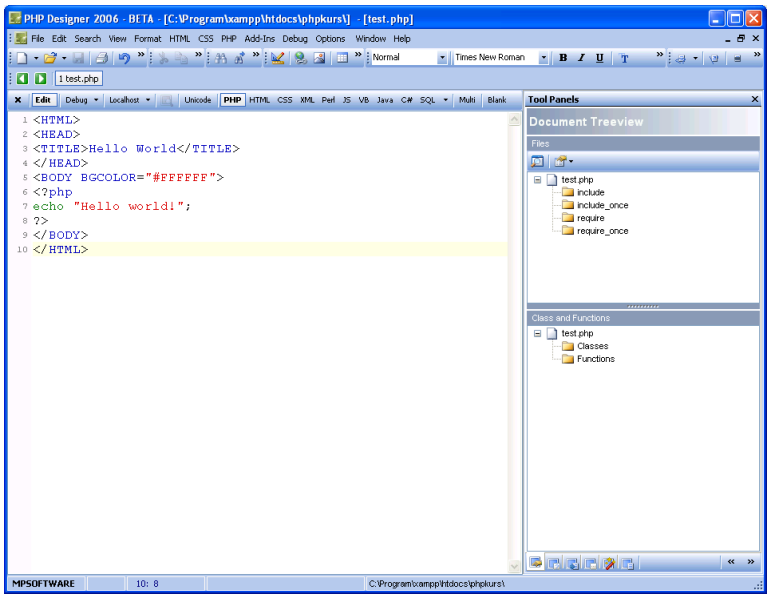
Under mappen **Debug** görs följande felsökningsinställningar:

PHP Interpreter	C:\Program\xampp\php\php-win.exe
Server path	http://localhost/
Local server path	C:\Program\xampp\htdocs\

I Mappen **Browsers** ställer vi in webbläsare för förhandsgranskning:

Internet Explorer	C:\Program\Internet Explorer\iexplore.exe
Firefox	C:\Program\Mozilla Firefox\firefox.exe

Redigeringsfönstret



I redigeringsfönstret skrivs programkoden in. Den får automatiskt färgkodning för att underlätta programmeringen.

Genom att växla flik till **Localhost** kan programkoden provköras. Ett klick på Fliken **Edit** växlar tillbaka till redigeringsläge.

Under menyn **HTML** finns snabbval för att lägga in bland annat formulärelement och tabeller. Menyn **Format** innehåller liknande HTML-märken för fetstil, stora och små rubriker med mera.

HTML och formulär

Grundstrukturen för en webbsida

Webbsidor består av kod i språket HTML - Hypertext Markup Language. En minimal webbsida kan se ut enligt följande:

```
<HTML>
<HEAD>
<TITLE> Titel </TITLE>
</HEAD>
<BODY>
text
</BODY>
</HTML>
```

HTML-koden består av märken (eng "tags"). Märkena inleds alltid med < och avslutas med >. De flesta skrivs i par med ett startmärke och ett slutmärke. Exempelvis är <HTML> och </HTML> startmärke respektive slutmärke och markerar starten och slutet på sidan. Texten mellan <TITLE> och </TITLE> bestämmer vad som ska stå i titelraden på webbläsaren. Märket <BODY> signalerar att innehållet i själva webbläsarefönstret tar vid. Några andra vanliga märken är
 för radbrytning. <P> och </P> som omsluter ett stycke, samt <H1> </H1> som skapar en stor rubrik.

För övriga HTML-märken finns en utmärkt guide från Umeå universitet på http://www.iml.umu.se/stod/internet/html/index.html

Formulär i HTML

Ett program eller ett skript är inte särskilt meningsfullt om det inte kan fråga efter information i från användaren.

I vanliga program matar användaren oftast in information från en kommandoprompt, eller vanligare nuförtiden, genom att fylla i en dialogruta eller välja ett menyalternativ. För att få in information från användaren till ett PHP-skript används formulär. Formulär ingår som en del i HTML-standarden.

Processen går alltså till på detta sätt:

1) Användaren fyller i ett formulär på en webbsida (HTML-dokument).
2) Formulärinformationen skickas till PHP-skriptet på webbservern.
3) PHP-skriptet behandlar informationen.
4) PHP-skriptet skriver ut en webbsida som användaren får som respons.

FORM-märket omsluter formuläret

Först ska vi titta på hur man utformar själva formuläret i HTML. Detta kan göras dels genom att koda för hand i ett redigeringsprogram, dels via ett grafiskt program som t.ex. Dreamweaver. Vi kommer att skriva koden för hand. Det första man behöver i ett formulär är två stycken HTML-märken som omsluter själva formuläret:

```
<FORM ACTION="phpskript.php" METHOD="POST">
</FORM>
```

ACTION talar här om sökvägen till det PHP-skript som ska ta hand om formulärinnehållet. Med METHOD specificerar vi med vilken metod formulärdata ska sändas iväg. Det finns två metoder: GET och POST. Används metoden GET skickas informationen med i adressen till php-skriptet. Denna metod används ofta av sökmotorer. Det kan t.ex. se ut så här: Man använder vanligen i stället POST, eftersom man med GET inte kan skicka mer än c:a 250 tecken. Med POST-metoden så paketeras informationen och skickas med m.h.a. HTTP-protokollet. Detta har också den fördelen att användaren inte ser all teknisk information som skickas - ett sätt att "gömma" information som skickas mellan webbsidorna.

<INPUT TYPE="text"> ger en vanlig textruta

Om vi matar in FORM-koden i föregående stycke, kommer ingenting alls att visas i webbläsaren. Vi behöver också ha rutor där användaren kan fylla i information. Ett vanligt textfält skrivs så här:

```
Ditt förnamn: <BR />
<INPUT TYPE="text" NAME="fornamn" VALUE="skriv här" SIZE="30">
```

Nästan alltid har man en beskrivning av formulärfältet till vänster, så att användaren vet vad som ska fyllas i. Sedan kommer själva HTML-märket. NAME måste finnas med - detta för att PHP senare ska kunna särskilja just detta fält och dess innehåll. När formuläret skickas iväg kommer nämligen automatiskt en variabel skapas med namnet $fornamn som innehåller det användaren har fyllt i detta fält. Observera att man ej bör ha svenska bokstäver i NAME-attributet. VALUE används oftast inte, men ger ett standardvärde i fältet. Till sist bestämmer vi hur många tecken brett formulärfältet ska vara med SIZE.

<INPUT TYPE="password"> - skapar lösenordsruta

Det här fältet fungerar precis som textrutan, med den skillnaden att det visas stjärnor när man fyller i fältet. Observera att informationen skickas helt i klartext till servern så ska man tillhandahålla känslig information med access via ett formulär är det lämpligt att skaffa en server med krypteringsfunktion.

Så här ser märket ut:

```
  Lösenord: <INPUT TYPE="password" NAME="losenord">
```

<INPUT TYPE="checkbox"> - ger en kryssruta

Det här fältet fungerar precis som kryssrutorna i t.ex. Windows - de kan antingen vara ikryssade eller ej ikryssade. Ett exempel:

```
<H2>Vilka semesterorter har du besökt</H2>
<INPUT TYPE="checkbox" NAME="azorerna" VALUE="1"> Azorerna<BR>
<INPUT TYPE="checkbox" NAME="mallorca" VALUE="1"> Mallorca<BR>
<INPUT TYPE="checkbox" NAME="teneriffa" VALUE="1"> Teneriffa
```

NAME talar om vilket namn variabeln kommer att få. Har användaren t.ex. kryssat i rutorna för Azorerna och Mallorca, kommer variabeln $azorerna få värdet 1, $mallorca också värdet 1, medan variabeln $teneriffa kommer att vara tom.

<INPUT TYPE="radio"> - skapa radioknappar

Det finns ytterligare en typ av knapp, nämligen radioknappar. Ni känner säkert igen denna från Windows också. Den används när man ska välja ett, och endast ett alternativ av flera i en grupp. T.ex. om man behöver skriva in åldern – en person kan ju inte gärna ha två åldrar samtidigt! Vi kan ju tänka oss följande exempel vid beställning av en färdbiljett:

```
<INPUT TYPE="radio" NAME="alder" VALUE="barn"> Under 16 år
<BR />
<INPUT TYPE="radio" NAME="alder" VALUE="ungdom"> 16 - 25 år
<BR />
<INPUT TYPE="radio" NAME="alder" VALUE="vuxen"> Över 25 år
```


-märkena i exemplet har vi med för att få ny rad. Observera att det här är viktigt att alla radioknappar i samma grupp (de man ska välja bland) har samma NAME-attribut. Hade vi gett dem olika namn hade vi kunnat kryssa i alla rutorna, vilket inte hade varit så bra. I stället särskiljer vi respektive fält med olika värden i VALUE. Om vi i exemplet fyllt i vuxen, så kommer PHP-skriptet att ta emot en variabel med namnet $alder och innehållet "vuxen".

<TEXTAREA> - flerradiga textfält

Flerradiga textfält används bland annat i webbmailtjänster som t.ex. Hotmail för att skriva in meddelandetexten. Då används TEXTAREA-fältet i HTML-koden. Observera att TEXTAREA har ett slutmärke (</TEXTAREA>). ROWS ställer in antalet rader, och COLS hur många tecken brett fältet ska vara.

```
<TEXTAREA NAME="meddelande" ROWS="20" COLS="40">
</TEXTAREA>
```

Andra formulärfält

Det finns några till, mer ovanliga formulärfält, bland annat "dropdown"-menyer. En snabbguide till HTML finns på adressen http://www.webmonkey.com/webmonkey/reference/html_cheatsheet/ och även där finns formulärelementen med.

<INPUT TYPE="submit"> - skicka iväg formuläret

Till sist behöver vi en knapp som skickar iväg formulärinnehållet till servern. Detta gör vi med en submit-knapp. Värdet i VALUE bestämmer vad som ska stå på knappen, men har ingen funktion i övrigt.

```
<INPUT TYPE="submit" VALUE="Skicka">
```

Komplett formulärexempel

Så är det dags att binda ihop allting och skapa ett komplett formulär. Vi börjar med ett enkelt exempel:

```
<HTML><HEAD><TITLE></TITLE></HEAD>
<BODY>
<FORM ACTION="phpskript.php" METHOD="POST">
<H1>Biljettbokning</H1>
<H2>Hur gammal är du</H2>
<INPUT TYPE="radio" NAME="alder" VALUE="barn"> Under 16 år <BR />
<INPUT TYPE="radio" NAME="alder" VALUE="ungdom"> 16 - 25 år <BR />
<INPUT TYPE="radio" NAME="alder" VALUE="vuxen"> Över 25 år <BR />
<INPUT TYPE="submit" VALUE="Skicka">
</FORM>
</BODY>
</HTML>
```

För att ta emot formulärinnehållet behöver vi också ett PHP-skript i "andra änden". Det kan till exempel se ut så här:

```
<HTML><HEAD><TITLE></TITLE></HEAD>
<BODY>
<?php
echo "Vänta medan priset för din " . $_REQUEST['alder'] . "-biljett räknas ut.";
?>
</BODY>
</HTML>
```

Alla formulärdata som skickats lagras i matrisen $_REQUEST. Hade formulärfältet i stället hetat namn skulle innehållet ha funnits i $_REQUEST['namn'].

Övningar HTML och formulär

Observera att dessa övningar endast innefattar HTML, i nästa övningsmoment ska du bygga på övningarna med kod i PHP som tar emot formulärinnehållet.

Övning 1.1

Gör ett formulär där användaren kan mata in varsitt tal i två stycken textrutor.

Övning 1.2

Gör ett komplett "kontaktformulär" med textrutor där du frågar efter namn, e-postadress och telefon. Du ska även skapa en kryssruta där användaren kan bocka i sitt intresse för ett nyhetsbrev. Till sist ska du skapa två stycken radioknappar, där användaren kan välja mellan att bli kontaktad per telefon eller e-post.

Övning 1.3

Skapa en sida där besökaren ska välja sin favoritfilm av tre olika filmer. Besökaren ska dessutom välja sin favoritbok bland tre olika böcker. Det ska gå att välja högst en bok och en film.

Övning 1.4

Utforma ett formulär på en sida där användaren ska mata in en temperatur i Celsius.

Övning 1.5

På din webbsida vill du ha ett formulär där besökaren kan lämna kommentarer. Kommentarerna ska kunna skrivas i ett flerradstextfält. Dessutom ska det finnas en vanlig textruta för att fylla i namn. På knappen för att skicka iväg formuläret ska det stå "Skicka kommentarer".

Variabler och kommentarer

Att skriva koden

Varje PHP-snutt börjar med <?php och avslutas med ?>. På detta sätt kan man väva in PHP-kod i vanliga HTML-dokument.

```
<HTML>
<HEAD>
<TITLE>Dagens datum</TITLE>
</HEAD>
<BODY>
</BODY>
</HTML>
```

Detta är ett helt vanligt HTML-dokument. Om vi tittar på det i en webbläsare kommer vi att få upp ett tomt dokument med titeln "Dagens datum". Nu kan vi väva in PHP-kod i dokumentet.

```
<HTML>
<HEAD>
<TITLE>Dagens datum</TITLE>
</HEAD>
<BODY>
<?php
echo date("Y-m-d H:i");
?>
</BODY>
</HTML>
```

Då får vi alltså automatiskt uträknat dagens datum för oss. Man kan även börja php-kod med endast <?, det är en smaksak vilket man väljer. "echo" i koden betyderatt vi ska skriva ut någonting på skärmen. Sedan använder vi "date" för att fådagens datum. Y talar om att vi vill ha årtalet (Year), m betyder månad och d dag imånaden. H står för hour vilket ger oss timslaget och i ger oss minuterna.Ett litet enklare exempel kan se ut så här:

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY>
<?php
echo "Hello World";
?>
</BODY>
</HTML>
```

Detta skriver ut raden "Hello World" på skärmen. Så fort man ska skriva ut någonting så måste det inneslutas i citationstecken, också kallade dubbelfnuttar. Observera också att varje rad avslutas med ett semikolon. Man kan välja att skriva ut HTML-kod inuti echo-satsen också:

```
<?php
echo "<B>Hello World</B>";
?>
```

Nu kommer texten som skrivs ut på skärmen att vara i fetstil (bold). Vi skulle kunna skriva in <B> märket som gör texten till fetstil i själva HTML-koden i stället.

```
<HTML>
<HEAD>
<TITLE></TITLE>
<BODY>
<B>
<?php
echo "Hello World";
?>
</B>
</BODY>
</HTML>
```

vilket ger samma resultat.

Variabler

En variabel är en behållare för ett värde. Tänk dig den som en kökslåda som innehåller någonting. Vi har ett namn på kökslådan dvs. själva variabeln och sedan har variabeln ett innehåll. Det finns två huvudtyper av variabler: strängar och tal. Strängar innehåller text medan tal innehåller siffror. Innan man använder en variabel behöver den tilldelas ett värde. Ett ex: (jag har här uteslutit HTML-koden runt skriptet för att vi ska koncentrera oss på det viktiga)

```
<?php
$namn = "Thomas Höjemo";
?>
```

Nu har variabeln $namn fått värdet "Thomas Höjemo". Observera att alla variabler alltid börjar med dollartecknet. Lika med tecknet talar om att variabeln ska få ett värde. När man ger en variabel ett strängvärde, vilket vi gör i detta fall, måste strängen vara omgiven av dubbelfnuttar. I annat fall kommer vi att få ett felmeddelande:

```
<?php
$namn = Thomas Höjemo;
?>
```

Parse error betyder att PHP helt enkelt inte förstod vad vi menade. Man brukar använda variabelnamn som har små bokstäver. Det är också viktigt att använda variabelnamn som gör det enkelt att förstå vad variabeln ska användas till, så det är bättre att kalla en variabel för t.ex. $efternamn än $var45. Variabelnamnet får innehålla bokstäver, siffror samt understreck (_) men kan ej börja med en siffra. Så hur använder man då variabler? Ofta vill man skriva ut variabelinnehållet på skärmen mixat med färdig text runt, och då gör man på följande sätt:

```
<?php
$namn = "Thomas Höjemo";
echo "$namn";
?>
```

I detta fall kommer alltså variabelinnehållet att skrivas ut i stället för $namn. Observera att detta endast gäller om dubbla citationstecken används. Vi kan se vad som händer om vi har enkla citationstecken (enkelfnuttar) i stället:

```
<?php
$namn = "Thomas Höjemo";
echo '$namn';
?>
```

Som ni ser skrivs $namn ordagrannt ut. Det är alltså viktigt att alltid använda dubbelfnuttar när man ska skriva ut variabler. Men det är ju tur att enkelfnuttar finns också, annars skulle det vara svårt att skriva ut dollartecken! Genom att använda punkttecknet kan man kombinera vanlig text och strängvariabler:

```
<?php
$namn = "Thomas Höjemo";
echo "Mitt namn är " . $namn . " och jag bor i Göteborg.";
?>
```

Vi kan även baka in variabeln mellan citationstecknen, men observera att detta endast går då det är mellanrum före och efter variabelnamnet:

```
<?php
$namn = "Thomas Höjemo";
echo "Mitt namn är $namn och jag bor i Göteborg.";
?>
```

För att använda tal i variabler så fungerar det på i stort sett samma sätt. Det är dock viktigt att komma ihåg att man aldrig ska ha några fnuttar runt talen. Observera också att man använder decimalpunkt, ej decimalkomma.

Ett exempel:

```
<?php
$tal1 = 37.76;
$tal2 = 45.23;
$summa = $tal1 + $tal2;
echo "Summan av $tal1 och $tal2 är $summa";
?>
```

Man kan använda + - * och / för att räkna med tal. Dessa kallas med ett finare uttryck för aritmetiska operatorer. Det finns också något som kallas logiska operatorer men de kommer vi att gå igenom senare.

Datatyper

PHP är ett löst typat språk. Detta innebär att man inte behöver deklarera vad för typ av variabel man vill ha innan man skapar den.

Det finns åtta olika typer av datatyper:

Datatyp	**Beskrivning**	**Förklaring**
Boolesk	1 eller 0	
Integer	heltal	positiva eller negativa heltal
Float	decimaltal	positiva eller negativa decimaltal
String	sträng	kan innehålla både bokstäver och siffror
Object	objekt	används vid objektorienterad programmering
Array	matris	innehåller flera variabler som var och en kan ha olika datatyp
Resource	resurs	en pekare till en extern resurs, t.ex. en databas
Null	ingenting	en variabel som ej har satts till något värde har värdet Null

Typkonvertering

PHP omvandlar tyst mellan olika variabeltyper

```
<?php
$tal1 = "7 kr";
$tal2 = "3 kr";
$summa = $tal1 + $tal2;
echo "Summan är $summa";
?>
```

I detta program sker typkonverteringen på raden där $tal1 och $tal2 adderas. Eftersom addition av förståeliga skäl är svårt att utföra med strängar, omvandlas variablerna $tal1 och $tal2 från strängar till tal. Slutresultatet blir att "Summan är 10" skrivs ut.

Det går även att explicit konvertera mellan olika typer. Detta görs genom att skriva typen man vill konvertera till enligt följande:

```
<?php
$antal = 100.32;
$antal = (int) $antal; // $antal innehåller heltalet 100
$antal = (double) $antal; // $antal innehåller decimaltalet 100 (100.0)
$antal = (string) $antal; // $antal innehåller strängen 100
?>
```

Detta kan bland annat vara användbart för att säkerställa att formulärdata verkligen är av rätt typ.

Kontrollera variabeltyp

Genom funktionen gettype kan man kontrollera vilken variabeltyp en viss variabel har.

```
<?php
$variabel = "Hej";
echo gettype($variabel); // Ger resultatet string
?>
```

För att undersöka om en variabel är av en viss typ används serien funktioner som börjar på is_. T.ex. is_int(), is_string(), is_double().

```
<?php
$variabel = "Hej";
if ( is_string($variabel) ) {
      echo "Det var en sträng!";
} else {
      echo "Det var inte en sträng!";
}
?>
```

Fördefinierade variabler

PHP har en stor mängd fördefinierade variabler. Det vanligaste användningsområdet för dessa är för att se vilken information som skickats från ett formulär. När ett formulär skickats iväg till ett PHP-skript så lagras automatiskt formulärelementens innehåll i matrisen $_REQUEST. Mer information om matriser kommer senare i kursen.

Har vi t.ex. skapat en textruta med namnet "adress" och användaren fyller i värdet "Karlsgatan 12" så kommer $_REQUEST['adress'] få innehållet "Karlsgatan 12". Fyller användaren i stället i textrutan "telefon" med numret "012-345678" får $_REQUEST['telefon'] innehållet "012-345678". För att skriva ut telefonnumret behövs endast echo enligt följande modell:

```
echo $_REQUEST['telefon'];
```

vilket kommer att resultera i att 012-345678 skrivs ut på skärmen förutsatt att användaren matade in just det numret i formuläret.

I det fall en inställning ändrats i PHP-systemkonfigurationen så kommer även fristående variabler skapas. I exemplen ovan skulle i så fall variablerna $adress och $telefon skapas. Denna inställning är avstängd från början, eftersom det kan medföra säkerhetsrisker om den används på ett slarvigt sätt.

$_SERVER['HTTP_REFERER'] innehåller den sida som besökaren kom ifrån till skriptet. (Notera att refererer skrivs med ett r, vilket beror på en pinsam felstavning i HTTP-protokollet.) $_SERVER['REMOTE_ADDR'] innehåller IP-adressen som besökaren har. $_SERVER['PHP_SELF'] innehåller den kompletta adressen till skriptet som körs. För mer information om alla fördefinierade variabler, se PHP-manualen på adressen http://www.php.net/manual/ .

Kommentarer

Genom att lägga in kommentarer i sitt skript kan man skriva vad ett speciellt avsnitt gör. Har man långa komplicerade skript kan detta vara bra, eftersom det annars kan vara svårt att förstå vad varje del är till för. Det finns två olika sätt att skriva kommentarer i PHP. Det första sättet ser ut så här:

```
<HTML>
<HEAD>
<TITLE></TITLE>
<BODY>
<?php
echo "Hello World"; // Skriver ut raden Hello World på skärmen
?>
</BODY>
</HTML>
```

// är avsett för kommentarer som sträcker sig över endast en rad. Med den andra typen av kommentarer kan man ha flera rader:

```
<HTML>
<HEAD>
<TITLE></TITLE>
<BODY>
<?php
/* Skript skrivet av
Thomas Höjemo
år 2006.
*/
echo "Hello World";
?>
</BODY>
</HTML>
Man börjar alltså kommentaren med /* och avslutar den med */.
```

Övningar variabler och kommentarer

Övning 2.1

Använd formuläret från övning 1.1. Skapa ett PHP-skript som tar emot de två talen, multiplicerar dem och presenterar resultatet så här: "Produkten av <tal1> och <tal2> är <summa>".

Övning 2.2

Använd formuläret från övning 2.2. Skapa ett skript som tar emot data från detta formulär: Skriptet ska skriva ut "Namn:" följt av namnet på personen, "E-post-adress:" och personens e-post-adress och till sist "Vi kommer att kontakta dig inom snarast per " följt av antingen e-post eller telefon beroende på vad användaren valt.

Övning 2.3

Gör ett formulär där användaren kan välja bland fyra färger att "måla" bakgrunden med för webbsidan som kommer upp när han/hon klickat på knappen Måla bakgrund. Skapa sedan ett skript som verkligen gör detta. Färgerna ska vara röd, blå, grön och gul.

Tips: färgerna skrivs med sina engelska namn enligt följande exempel (som visar en sida med vit bakgrund):

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="white">
</BODY>
</HTML>
```

Övning 2.4

Skriv ett program som tar en siffra (från formuläret i övning 1.4) som innehåller dagens temperatur i Celsius. Programmet ska sedan skriva ut hur många grader Fahrenheit det motsvarar enligt följande mall: "100 grader Celsius motsvarar 212 grader Fahrenheit". Formeln för omvandlingen är F = (9/5)*C + 32 där F står för grader Fahrenheit och C för grader Celsius.

Övning 2.5

Försök att lista ut vad följande program kommer att skriva ut.

```
<?php
$ett = '$noll';
echo "$ett";
echo '$ett';
?>
```

Prova sedan att mata in programmet och se om du gissade rätt.

Operatorer, villkorssatser och loopar

Operatorer

Det finns ett antal olika operatortyper i PHP. Från matematiken känner vi igen de aritmetiska operatorerna, t.ex. +,-,* och /. Dessa har vi tittat på tidigare. Tilldelningsoperatorn kallas likamed-tecknet. Den har vi också använt oss av för att ge en variabel ett värde. Några exempel:

```
<?php
$a = 324; // Ger $a värdet 324
$a = a + 1; // Ökar $a med 1 till 325
$a = berakna(12); // $a får returvärdet av funktionen berakna
?>
```

Ett ytterligare exempel på en operator är strängkonkateneringsoperatorn (rekordlångt namn). Den består av en punkt och gör med strängar vad + gör med siffror:

```
<?php
$namn = "Thomas";
$a = "Hej " . $namn;
?>
```

Två andra vanliga operatorer är villkorsoperatorer och logiska operatorer. De används ofta tillsammans, vilket vi kommer att se nu.

Villkorssatser

Hittills har alla program vi skrivit körts igenom rad för rad, utan att vi kunnat styra in programmet på olika vägar beroende på vad användaren har matat in. Det är detta vi har styrstrukturer till. Styrstrukturer har det gemensamt att de kontrollerar om ett uttryck stämmer med ett annat och sedan beroende på resultatet gör ett val. På detta sätt kan man skriva mycket mer flexibla program. Det finns två huvudtyper av styrstrukturer: villkorssatser och loopar. Vi börjar med villkorssatser.

```
<?php
if ($losenord == "test123") {
      echo "Du är inloggad";
}
?>
```

I detta lilla skript använder vi en villkorssats, eller if-sats som den också heter. If-satsen är sig lik från nästan alla programmeringsspråk och är nödvändig för att man ska anpassa vad som händer efter vad användaren gör. Det som står innanför paranteserna är det s.k. jämförelseuttrycket. I detta fall kontrollerar vi om variabeln $losenord har innehållet "test123". Observera att vi använder två likamedtecken! Hade vi bara använt ett likamedtecken hade vi i stället tilldelat variabeln $losenord värdet test123.

Om lösenordet stämmer, körs allting som står innan för krullparanteserna. Om lösenordet inte stämmer, fortsätter programmet efter högerkrullparantesen. Vi kan också välja att ha med ett alternativ som ska inträffa då lösenordet inte stämmer. Då kan det se ut så här:

```
<?php
if ($losenord == "test123") {
echo "Du är inloggad";
} else {
echo "Fel lösenord";
}
?>
```

I det här fallet kommer vi alltså att få meddelandet "Fel lösenord" om lösenordet inte stämmer.

Det finns fler jämförelseoperatorer än ==, även om == är den överlägset vanligaste. Några jämförelseoperatorer finns listade i boken på sidan 126. Vi har bland annat < som står för mindre än, > som står för större än. Dessa två används endast för tal. != står för inte lika med och kan däremot användas både för tal och strängar. Man kan bygga på if-satser med elseif. På detta sätt kan programmet ta fler än två olika vägar.

```
<?php
if ($veckodag == "lördag") {
      echo "Det är lördag och jag kan ta det lugnt.";
} elseif ($veckodag == "söndag") {
      echo "Söndag är vilodag.";
} else {
      echo "Vanlig vardag";
}
?>
```

Man kan även testa att två olika villkor i samma if-sats. Detta kan göras med and eller med or. Använder man and måste båda villkoren stämma. Med or räcker det att minst ett av dem stämmer. En tredje variant är xor eller exklusivt eller, då måste antingen det ena eller det andra villkoret stämma. En lista på alla logiska operatorer, som dessa kallas, finns på sidan 127.

```
<?php
if ($anvandarnamn == "guest" and $losenord ="test123") {
      echo "Du är inloggad";
} else {
      echo "Fel användarnamn eller lösenord";
}
?>
```

Loopar

Nu vidare till nästa variant på styrstrukturer, nämligen loopar. Den första varianten vi ska titta på är en for-loop.

```
<?php
for ($i = 1;$i < 20;$i = $i + 1) {
      echo "Detta är rad $i <BR>";
}
?>
```

I detta exempel kommer vi att få 19 rader utskrivna. For-loopen har tre delar, det första kallas initiering, där vi sätter loopvariabeln till ett visst värde. I vårt fall har vi valt att ha $i som loopvariabel. Den andra delen kallas villkorssats, och den testas varje gång loopen körts. Den tredje satsen kallas uppräkningssats, och innebär att loopvariabeln räknas upp med 1 efter varje varv.

While loopen är en mer generell variant av for-loopen. Den används för att upprepa något så länge ett visst villkor är uppfyllt. Motsvarande kod skulle alltså se ut så här:

```
<?php
$i = 1;
while ($i < 20) {
echo "Detta är rad $i <BR>";
$i++;
}
?>
```

Inuti for och while loopar kan man använda break för att omedelbart avbryta körningen av loopen och fortsätta direkt med koden som kommer efter loopens slut. Med continue kan man fortsätta direkt till nästa varv av loopen.

Övningar operatorer, villkorssatser och loopar

Övning 3.1

Modifiera temperaturkonverteringsprogrammet ytterligare en gång, så att användaren matar in en temperatur i ett textfält, och sedan väljer om konverteringen ska ske från F till C eller från C till F med hjälp av radioknappar.

Övning 3.2

Gör ett skript som frågar efter användarnamn och lösenord via ett formulär. Om användarnamnet och lösenordet är korrekt ska en sida visas där det står "Du är inloggad". Annars ska användaren få upp inloggningsrutan igen.

Övning 3.3

Skriv ett program som frågar efter två heltal via ett formulär, det första talet ska vara lägre än det andra. Skriv ut alla heltal mellan de två som matats in. Separera med mellanslag.

Övning 3.4

Gör ett program som skriver ut tal och respektive tals kvadrat från 0 till 50.

Övning 3.5

Gör ett program som är en lånekalkylator. M.h.a. radioknappar ska användaren kunna välja mellan 1, 3 och 5 års lånetid. I en ruta ska användaren skriva i lånebeloppet och i nästa räntan i hela procent. Programmet ska sedan räkna ut den totala lånekostnaden (Räknas ut genom ränta på ränta-principen, årsvis). Så för ett tvåårigt lån på 5000 med räntan 4% skulle alltså lånekostnaden bli 5000*1,04*1,04 - 5000. Observera att lånet är "amorteringsfritt".

Funktioner

Funktioner fungerar ungefär som ett snabbkommando för att göra en lång lista av kommandon i ett svep. Därmed slipper man upprepa kod i programmet, vilket gör det lättare att skriva och uppdatera. Förutom de många funktioner som PHP har inbyggt, kan man alltså skapa egna funktioner. Följande exempel skriver ut en början på ett HTML-dokument:

```
<?php
function starthtml() {
      echo "<HTML><HEAD><TITLE></TITLE></HEAD><BODY>";
}
starthtml();
echo "testdokument";
echo "</BODY></HTML>";
?>
```

Det första vi gör är att deklarera funktionen. Detta gör vi med hjälp av function följt av namnet på funktionen och sedan två paranteser. Längre ned i koden anropar vi funktionen. Detta gör vi genom att skriva funktionsnamnet följt av två paranteser.

Skicka argument

Vi kan modifiera funktionen så att den tar emot argument. Argument skickas till funktionen och används sedan på något sätt inuti funktionen.

```
<?php
function starthtml($title) {
      echo "<HTML><HEAD><TITLE>$title</TITLE></HEAD><BODY>";
}
starthtml("Test");
echo "testdokument";
echo "</BODY></HTML>";

?>
```

Nu har vi alltså gjort funktionen en aning mer flexibel, genom att vi kan ändra titeln på sidan. Vi skickar med strängen med innehållet "Test", denna läggs sedan i variabeln $title efter funktionsanropet. Sedan kan funktionen använda denna variabel, i det här fallet för utskrift.

Standardvärde

En funktion kan definieras så att den har ett standardvärde, som används om inget värde anges vid funktionsanropet.

```
<?php
function fargad_text($text = "exempeltext", $farg = "green") {
      echo "<FONT COLOR='$farg'> $text </FONT>";
}
?>
```

I exemplet ovan tar funktionen fargad_text två argument. $text har standardvärdet "exempeltext" och $farg standardvärdet "green". Om man anropar funktionen utan några argument kommer standardvärdena att användas.

```
fargad_text("annan text");
```

Skickas endast ett argument används det för $text och $farg behåller sitt standardvärde.

Returnera värde

Funktioner kan också returnera ett värde. Låt oss titta på ett exempel på detta.

```
<?php
function area_cirkel($radie) {
      $area = $radie * $radie * M_PI;
      return $area;
}
echo area_cirkel(10);
?>
```

I och med att vi använder return skickas variabeln $omkrets innehåll tillbaka till det ställe som funktionen anropades från. (M_PI är en inbyggd konstant som innehåller ett värde av pi med många decimaler.)

Lokal räckvidd

Det är viktigt att notera att variabler skapade inuti funktioner ej existerar utanför funktionens krullparanteser. Detta brukar på engelska kallas "local scope" d.v.s. lokal räckvidd. Se följande exempel:

```
function berakna_summa($a,$b) {
      $summa = $a + $b;
      $c = 10;
      return $summa;
}
$resultat = berakna_summa($a,$b);
print $c;
```

Det första vi gör är att skapa funktionen berakna_summa. Denna funktion tar två argument, $a och $b. Sedan lägger den ihop dessa två argument och stoppar i variabeln $summa. En ny variabel vid namn $c skapas på nästa rad. Men eftersom $c skapas inuti funktionen blir det en lokal variabel. Den existerar bara inom funktionens krullparanteser. När vi senare på sista raden försöker skriva ut variabeln $c kommer det inte att synas någonting på skärmen eftersom $c inte existerar utanför funktionen.

```
function dubbla($a) {
      $a = $a * 2;
      global $c = 10;
      return $a;
}
$resultat = dubbla($a);
print $c;
```

I detta nya exempel kommer däremot 10 att skrivas ut. Detta eftersom vi talat om för PHP att $c ska vara en global variabel, som även existerar utanför funktionen den skapats i.

Inbyggda funktioner

PHP har massor med användbara inbyggda funktioner, som vi ska titta närmare på senare. En utmärkt referens till alla inbyggda funktioner är PHP-manualen, som finns på adressen http://www.php.net/manual/ .

Övningar funktioner

Övning 4.1

Skriv en funktion som tar en variabel som argument och skriver ut följande på skärmen: Du skrev in variabeln: följt av innehållet i variabeln. Testa sedan funktionen genom att skicka 456 i en variabel till den.

Övning 4.2

Skriv en funktion som automatiskt returnerar argumentet med HTML-koder för fetstil och typsnittet Arial runt. Ex: Om man skickar "Rubrik" så ska funktionen skicka tillbaka <B><FONT FACE="Arial">Rubrik</FONT></B>

Övning 4.3

Skapa en funktion som tar två argument och skriver ut deras sammanlagda antal tecken på skärmen. Längden på en sträng fås genom den inbyggda funktionen strlen.

Övning 4.4

Skapa en funktion som räknar ut omkretsen på en cirkel. Funktionen ska ta radien som ett argument. Definiera pi som en konstant i början av programmet, och använd 3,14 som ungefärligt mått på pi. Som en extrauppgift så avrunda värdet till närmaste heltal. Detta görs med den inbyggda funktionen round.

Övning 4.5

Skapa ett program med en funktion (print_table_start) som skriver början på en tabell i HTML och tar ett argument i form av färgen på tabellen (red, blue et.c.). Skapa en annan funktion som heter print_table_row och skriver ut en tabellrad och tar två argument, det som ska stå i första tabellcellen och det som ska stå i andra tabellcellen. Till sist skapar du en funktion kallad print_table_end som skriver ut slutet på tabellen.

Ex på en tabell i HTML: <TABLE> talar om att tabellen börjar. TR står för tabellrad, och TD för tabellcell.

```
<HTML>
<HEAD><TITLE>titel på sidan</TITLE></HEAD>
<BODY>
<TABLE BGCOLOR="red">
<TR><TD>cell 1 </TD> <TD> cell 2 </TD></TR>
<TR><TD>cell 3 </TD> <TD> cell 4 </TD></TR>
</TABLE>
</BODY>
</HTML>
```

Matriser och foreach-loopen

En matris är en samling av variabler, tänk er ett helt skåp med kökslådor. Matrisen är då hela skåpet och innehåller flera kökslådor. I varje kökslåda finns ett innehåll d.v.s. ett värde. I de flesta programmeringsspråk så börjar man av någon outgrundlig anledning alltid att räkna från och med 0 i stället för 1. Detta innebär att den första "lådan" / indexet är 0, den andra 1 och så vidare. För att skapa den här matrisen i form av en kökslåda i PHP skulle vi göra så här:

```
<?php
$koksskap[0] = "bestick";
$koksskap[1] = "servetter";
$koksskap[2] = "påsar";
?>
```

Vi fyller alltså helt enkelt "lådorna" en i taget med innehåll. Vi behöver inte ens ta med siffrorna, utan kan låta PHP räkna ut dessa åt oss:

```
<?php
$koksskap[] = "bestick";
$koksskap[] = "servetter";
$koksskap[] = "påsar";
?>
```

Detta ger alltså samma resultat. För att sedan komma åt innehållet i respektive låda (värdet) så gör vi så här:

```
<?php
$koksskap[0] = "bestick";
$koksskap[1] = "servetter";
$koksskap[2] = "påsar";
echo "Översta lådan innehåller $koksskap[0] <BR>";
echo "Mittenlådan innehåller $koksskap[1] <BR>";
echo "Understa lådan innehåller $koksskap[2] <BR>";
?>
```


-märket betyder Break och är en HTML-kod för att byta till ny rad.

Man kan i stället för att använda 0,1,2 ge strängnamn åt indexen.

```
<?php
$koksskap["topp"] = "bestick";
$koksskap["mitten"] = "servetter";
$koksskap["botten"] = "påsar";
echo "Översta lådan innehåller $koksskap["topp"] <BR>";
echo "Mittenlådan innehåller $koksskap["mitten"] <BR>";
echo "Understa lådan innehåller $koksskap["botten"] <BR>";
?>
```

När använder man sig då av matriser? Man skulle ju i princip lika gärna kunna använda sig av variabler var och en för sig. Det finns några olika anledningar. Bland annat kan man på ett enkelt sätt sortera i matriser. Man kan också enkelt systematiskt gå igenom en matris och använda alla värdena. När man använder sig av funktioner kommer ofta returvärdet i form av en matris.

Genom unset kan man ta bort element ur matriser.

```
$namn[0] = "Gult";
$namn[1] = "Blått;
unset($namn[0]);
```

Nu togs alltså elementet Gult med nyckeln / indexet 0 bort ur matrisen.

foreach-loopen

Tillsammans med matriser används ofta foreach-loopen. Den går igenom matrisen ett index i taget.

```
<?php
$domains['se'] = "Sverige";
$domains['de'] = "Tyskland";
$domains['no'] = "Norge";
foreach ($domains as $index => $varde) {
      echo "$index är landskod för $varde <BR>";
}
?>
```

Detta exempel skriver ut följande på skärmen:

```
se är landskod för Sverige
de är landskod för Tyskland
no är landskod för Norge
```

Sortera matriser

Funktionen sort sorterar matriser efter deras värde, se följande exempel:

```
<?php
$tal[] = 321;
$tal[] = 3;
$tal[] = 7;
sort($tal);
foreach ($tal as $a) {
      echo $a . "<BR>";
}
?>
```

Detta exempel sorterar helt enkelt talen i nummerordning och skriver sedan ut dem i tur och ordning. Fler exempel på sorteringsfunktioner finns i boken i kapitlet om matriser.

Dela upp variabler med explode

Med hjälp av explode kan man dela upp en vanlig variabel i bitar och "stoppa in" bitarna i en matris. Exempel:

```
$text = "detta är en testtext.";
$pattern = " ";
$ord = explode($pattern, $text);
echo "$ord[2] $ord[3] $ord[1] $ord[0]";
```

I exemplet delas först upp innehållet i variabeln text i delar med mellanslag som skiljetecken. Inuti ord kommer först på position 0 att finnas "detta", därefter "är" på position 1 o.s.v.

Slå samman en matris med implode

Med implode så gör vi tvärtom - slår samman delarna i en matris till en vanlig sträng.

```
$ord[0] = "detta";
$ord[1] = "är";
$ord[2] = "en";
$ord[3] = "variabel";
echo implode(" ", $ord);
```

I det här fallet så slås alla delarna i matrisen $ord ihop, ett mellanslag läggs i varje sammanfogning och resultatet skrivs ut.

Skapa matriser automatiskt i formulär

Genom att avsluta namnet på till exempel en textruta med hakparanteser, exempelvis namn[] så skapas automatiskt en motsvarande matris i PHP.

```
<FORM ACTION="phpskript.php" METHOD="post">
<INPUT TYPE="text" NAME="namn[]">
<INPUT TYPE="text" NAME="namn[]">
<INPUT TYPE="submit">
</FORM>
```

I det här fallet får $_REQUEST['namn'][0] innehållet som skrivits i den första textrutan och $_REQUEST['namn'][1] innehållet i den andra textrutan.

Övningar matriser och foreach-loopen

Övning 5.1

Skapa ett formulär där användaren matar in 10 st namn. Kalla textrutorna för namn1, namn2 et.c. Sortera sedan namnen och skriv ut dem i bokstavsordning, ett på varje rad.

Övning 5.2

Utveckla programmet i övning 1 så att det skriver ut namnen i en tabell med ett radnummer först i tabellraden. Tabellraderna ska ha alternerande färger, (den första raden i grått den andra i vitt et.c.)

Övning 5.3

Skriv en funktion som tar ett tal mellan 1 och 9 som ett argument och sedan returnerar det svenska namnet för talet (ett, två, tre etc). Om talet är större än 9 så returnerar du i stället talet som vanligt (t .ex. 11). Testa att subrutinen fungerar som tänkt.

Övning 5.4

Fortsätt med programmet i övning 3 och utöka programmet så att funktionen tar två siffror, lägger ihop dem och presenterar resultatet i bokstavsform. (Ex: fyra plus tre blir sju.)

Stränghantering

Stränghantering används ofta i samband med inmatning för användare. Ibland för att kontrollera så att en e-post-adress är korrekt utformad, och ofta dessutom av säkerhetsskäl.

Ta bort mellanrum i början och slutet - trim

Ibland kan användare av misstag råkat skriva extra mellanrum efter inmatningar av t.ex. adress eller telefonnummer. När man sedan sparar data i t.ex. en databas, vill man inte ha med mellanrummen. Därför använder man funktionen trim för att ta bort dem. Observera att detta enbart gäller mellanrum precis i början eller slutet av en sträng; aldrig mitt i. Funktionen tar förutom mellanslag även bort nyradtecken och tabtecken. Om vi t.ex. har en variabel med namnet $adress kan vi göra så här:

```
$adress = trim($adress);
```

Omvandla till små eller stora bokstäver - strtolower respektive strtoupper

När man ska jämföra strängar tar PHP som standard hänsyn till små och stora bokstäver. Om $inmatat har värdet "Test" och $kontroll värdet "test" så kommer en jämförelse mellan dem med hjälp av == ej vara sann. Men om man då inte vill ta hänsyn till små och stora bokstäver? Då kan man använda strtolower eller strtoupper. Om vi tar exemplet med jämförelsen av $a och $b:

```
$kontroll = "test";
if (strtolower($inmatat) == $kontroll) {
      // gör något...
} else {
      // gör något annat...

}
```

Observera att $inmatat fortfarande har värdet "Test". Vill vi förändra detta gör vi så här:

```
$inmatat = strtolower($inmatat);
```

Funktionen strtoupper fungerar på samma sätt, med skillnaden att alla bokstäver blir versaler.

Längden på en sträng - strlen

Med strlen (står för string length) får vi reda på antalet tecken i en sträng.

```
$text = "Hej och hopp";
echo strlen($text); // Skriver ut 12
```

Plocka ut delar av en sträng - substr

Följande exempel tar ut de tre första tecknen ur strängen $text:

```
$text = "Hej och hopp";
$start = substr($text,0,3); // Plockar ut "Hej";
```

Observera att det första tecknet har position 0! Först specificerar vi namnet på variabeln, sedan positionen och sist längden på den delsträng vi vill ha ut. Det går även att ta delsträngar från slutet:

```
$text = "Hej och hopp";
$slut = substr($text,-4);// Plockar ut "hopp";
```

Här får vi alltså ut de sista fyra tecknen i strängen.

Söka igenom en sträng - strstr

Ibland vill man ta reda på om en viss text finns inuti en sträng. Detta gör man med strstr:

```
$namn = "Ulrika Eriksson";
$efternamn = strstr($namn, " "):
```

Här söker vi efter strängen " " det vill säga ett mellanslag. Eftersom vi hittar ett mellanslag i strängen $namn, kommer nu resten av strängen efter mellanslaget returneras.

Det går även att använda strstr i en if-sats för att köra olika kod beroende på om strängen hittades eller inte:

```
$namn = "Ulrika Eriksson";
if (strstr($namn, " ")) {
      echo 'Variabeln $namn innehåller mellanslag';
} else {
      echo 'Variabeln $namn innehåller inte mellanslag';
}
```

Söka och ersätta - str_replace

Funktionen str_replace används för att söka efter en delsträng och ersätta den med något annat.

```
$ny_variabel = str_replace("vit", "svart", "vit katt på taket");
echo $ny_variabel;
```

I detta fall kommer texten "svart katt på taket" skrivas ut.

Övningar stränghantering

Övning 6.1

Gör ett formulär där användaren ska fylla i namn, adress, postnr och postort. Kontrollera att alla fälten är ifyllda, och innehåller minst 3 tecken. Kontrollera att postnumret innehåller 5 tecken och att de tecknen endast är siffror.

Övning 6.2

Bygg på formuläret så att användaren också ska fylla i en e-post-adress. Kontrollera sedan att e-post-adressen innehåller ett @, och minst en punkt. Kontrollera också att e-post-adressen är minst sex tecken lång.

Övning 6.3

Utveckla skriptet i övning 6.2 så att det tar bort mellanslag i postnumret och därmed tillåter postnummer inskrivna enligt formen "415 22".

Reguljära uttryck

Med reguljära uttryck kan man göra en s.k. mönstermatchning av ett uttryck. Reguljära uttryck tillhör de svåraste områdena av PHP. I kapitlet om strängar i boken finns en utförligare beskrivning av reguljära uttryck. De används ofta för att göra en kontroll av inkommande formulärdata för att säkerställa att dessa håller sig inom vissa givna ramar och därigenom säkra programmet från angrepp och fel.

Det finns två olika typer av reguljära uttryck i PHP, POSIX-kompatibla och Perl-kompatibla. I grundsyntaxen är de ganska lika men de Perl-kompatibla reguljära uttrycken har fler avancerade funktioner. I PHP-manualen på adressen http://www.php.net/manual/en/pcre.pattern.syntax.php finns en detaljerad guide. Vi börjar med ett enkelt exempel:

```
$text = "Test 123";
if (preg_match("/123/",$text)) {
      echo 'Variabeln $text innehåller 123';
} else {
      echo 'Variabeln $text innehåller inte 123';
}
```

Funktionen preg används för mönstermatchning. Sökbegreppet står alltid mellan snedstreck. I exemplet söker vi efter teckenföljden 123 i strängen $text.

Matcha vissa tecken

```
$text = "Test 123";
if (preg_match("/[a-zåäö]/",$text)) {
      echo 'Variabeln $text innehåller minst en bokstav';
} else {
      echo 'Variabeln $text innehåller inte några bokstäver';
}
```

Genom att använda hakparanteser kan vi specificera en följd av tecken i mönstret. Strängen kommer att matchas om det finns minst en av de gemena (små) bokstäverna a-z, å, ä eller ö i den.

Negativ matchning

```
$text = "Test 123";
if (preg_match("/[^0-9]/",$text)) {
      echo "Variabeln innehåller minst ett tecken som inte är en siffra";
}
```

Genom att sätta ^ direkt till höger om hakparantesen inverteras mönstret.

Matcha oberoende av små och stora bokstäver

```
$text = "Test 123";
if (preg_match("/[a-zåäö]/i",$text)) {
      echo 'Variabeln $text innehåller minst en bokstav';
} else {
      echo 'Variabeln $text innehåller inte några bokstäver';
}
```

Genom växeln /i som står för case-insensitive så matchar vi både små och stora bokstäver.

Matcha ett visst antal av ett mönster

Genom specialtecknen ?, * och + kan vi matcha 0 eller 1, 0 eller flera respektive 1 eller flera av ett visst mönster.

```
$text = "abbc";
if (preg_match("/ab+c/", $text)) {
      echo "Innehåller ett a följt av ett eller flera b följt av ett c";
}
```

Med + matchar vi ett eller flera b:n enligt ovan.

```
$text = "abbc";
if (preg_match("/ab{1,2}c/", $text)) {
      echo "Innehåller ett a följt av ett eller två b följt av ett c";
}
```

Det går också att specificera exakt antal av ett visst tecken. Det gör man genom att skriva {x,y} där x står för minsta antal och y för största antal.

Matcha från början till slutet av en sträng

Hur gör vi då om vi ska kontrollera att strängen endast innehåller bokstäver och inget annat?

```
$text = "Test 123";
if (preg_match("/^[a-zåäöA-ZÅÄÖ]+$/",$text)) {
      echo 'Variabeln $text innehåller bara bokstäver';
} else {
      echo 'Variabeln $text innehåller inte bara bokstäver';
}
```

Här ser vi några små förändringar. Tecknet ^ står för "början av strängen" när det kommer som första tecken inuti mönstret. Detta innebär att strängen måste börja med en bokstav. Tecknet + står för "en eller flera av föregående mönster". Till sist kommer $ som står för "slutet av strängen" när den är inskriven på sista platsen i mönstret.

Observera att + måste vara med annars matchar vi bara strängar med endast ett tecken! Genom växeln /m kan man använda ^ och $ rad för rad i strängen. Detta innebär att mönstret endast behöver matcha minst en rad i stället för hela strängen.

Alternativmönster

```
$text = "gurka";
if (preg_match("/banan|gurka/", $text)) {
      echo 'Variabeln $text innehåller banan eller gurka';
} else {
      echo 'Variabeln $text innehåller varken banan eller gurka';
}
```

Specialtecknet | betyder eller - matcha ett visst mönster eller ett annat.

Delmönster

Genom att använda paranteser kan man gruppera mönstret i delmönster.

```
$text = "använd";
if (preg_match("/använd(armanual|ningsområde|)/", $text)) {
      echo "Matchar använd, användarmanual, användningsområde";
}
```

Mönstret matchar använd, eftersom det är tomt till höger om den sista |.

Byta ut text mot en annan text

Funktionen preg_replace byter ut en viss text mot en annan.

```
$text = 'Min e-post-adress är thohoj02@student.chalmers.se';
$pattern = 'thohoj02@student.chalmers.se';
$replace = "thomas@snt.se";
$nytext = preg_replace("/$pattern/", $replace, $text);
echo $nytext;
```

Det första argumentet är mönstret som ska sökas efter, den andra strängen som det ska ersättas med och det tredje variabeln som ska sökas igenom. Som returnerat värde från kommer variabeln med innehållet ersatt enligt önskemålet.

Lathund för reguljära uttryck

\	generellt escape-tecken
\n	matchar ny rad-tecken – newline
\t	matchar tabb-tecken
\d	siffra - samma sak som [0-9]
\D	inte siffra - samma sak som [^0-9]
\s	matchar mellanrumstecken - mellanslag, ny rad, tabb, formfeed, return
\S	matchar allt utom mellanrumstecken
^	matchar från början av sträng (eller början av rad med växeln /m)
\A	matchar från början av sträng (oberoende av växeln /m)
$	matchar till slutet av sträng (eller slutet av rad med växeln /m)
\Z	matchar till slutet av sträng (oberoende av växeln /m)
.	matchar vilket tecken som helst för utom ny rad-tecken (\n)
[a-z]	teckenföljd
|	logiskt "eller"
(x)	delmönster
?	matchar 0 eller 1 av föregående mönster
*	matchar 0 eller fler av föregående mönster
+	matchar 1 eller fler av föregående mönster
{x,y}	matchar mellan x och y gånger av föregående mönster

Övningar reguljära uttryck

Övning 7.1

Gör ett formulär där användaren ska fylla i ett domännamn. Kontrollera sedan att domännamnet slutar på .com, .net eller .org. Du ska också kontrollera att de övriga tecknen endast består av bokstäver a-z, siffror 0-9 eller bindestreck (-). Första tecknet måste vara en bokstav. Domännamnet ska vara minst sex tecken och högst 200 tecken långt.

Övning 7.2

Konstruera ett reguljärt uttryck som matchar en sträng som innehåller ett "t" följt av ett eller två "o". Endast små bokstäver ska matchas.

Övning 7.3

Konstruera ett reguljärt uttryck som matchar en sträng som börjar med "Det var en gång". Det spelar ingen roll om det första d:et är stort eller litet.

Övning 7.4

Konstruera ett reguljärt uttryck som ska kontrollera adresser som ska föras in i en databas. Adresserna får endast bestå av små och stora bokstäver, punkt, siffror och mellanslag.

Skicka e-post

SMTP-protokollet skickar e-post

SMTP står för Simple Mail Transfer Protocol och är protokollet som används varje gång man skickar iväg ett e-post-meddelande. Det finns flera serverprogramvaror som är gjorda för att hantera ivägskickandet av e-post. Sendmail, Postfix och Exim hör till de vanligaste SMTP-servrarna. För att PHP ska kunna e-post till SMTP-servern behöver vi tala om sökvägen till servern i php.ini:

```
sendmail_path = /usr/bin/sendmail
```

Använder du Windows följer oftast inte någon SMTP-server med. Då behöver du ställa in den separata SMTP-server som PHP ska ta kontakt med - enklast är att använda din internetleverantörs SMTP-server. Vilken adress det är varierar, exempelvis är adressen smtprelay1.telia.com till Telias SMTP-server.

```
SMTP = smtprelay1.telia.com;
sendmail_from = 'dinmailadress@telia.com>'
```

E-posten förs över helt som text, detta för att bibehålla kompabilitet med gamla system. Före själva meddelandet kommer meddelanderubrikerna. Av dessa är tre stycken obligatoriska:

From: Talar om vem som skickade e-posten: ex: <thomas@snt.se>
To: Talar om till vem meddelandet ska
Date: Talar om när e-post-meddelandet skickades

En fjärde används nästan alltid men är inte obligatorisk:

Subject: Vad e-post-meddelandet handlar om

Förutom dessa rubriker finns ett antal andra meddelanderubriker som brukar förekomma i e-post, de finns listade i boken "PHP Programmering" på sidan 261.

Funktionen mail

PHP har en enkel funktion för att skicka e-post, som fyndigt nog heter mail. Den används på detta sätt:

```
$amne = 'Viktigt meddelande';
$mottagare = 'test@test.se';
$avsandare = "From: thomas@snt.se\n";
$meddelandetext = "Testar att skicka e-post\n med PHP";
mail($mottagare,$amne,$meddelandetext,$avsandare);
```

PHPMailer

För alla utom de allra enklaste fallen är det bättre att använda sig av ett tilläggsbibliotek som heter PHPMailer. Det finns att ladda ned på adressen http://phpmailer.sourceforge.net/ där det också finns utförliga exempel på användandet.

Övningar skicka e-post

Övning 8.1

Gör ett formulär med fält för avsändaradress, adress e-posten ska skickas till, själva meddelandet och ärende. Skriv sedan ett PHP-program som tar hand om formuläret och skickar e-posten.

Övning 8.2

Utveckla programmet i övning 1 så att man kan skriva in flera adresser separerade med kommatecken. Programmet ska sedan skicka e-post-meddelandet till alla adresserna. Tips: använd funktionen split.

Filhantering

Innan vi går in på databashantering, så är det bra att känna till att det finns en enklare metod att spara data; att helt enkelt skriva dem till en fil på servern. Denna metod lämpar sig bäst när behöver en lagra mindre mängd data och inte har något behov av att sortera eller välja ut en delmängd av det sparade.

För att öppna en fil i PHP används fopen. Observera att vi måste ha rättighet att öppna filen. På UNIX och Linux-system finns ett rättighetssystem där man med hjälp av kommandot chmod ställer in vilka som har rättigheter att läsa och skriva filer. Hur rättigheterna fungerar finns beskrivet i boken "PHP Programmering" på s. 196. Just nu räcker det att känna till att ni behöver skriva

```
chmod ugo+r filnamn
```

för att PHP ska kunna läsa filen. Det finns olika sätt att öppna en fil, vi kommer först att titta på möjligheten att öppna en fil för läsning. En komplett beskrivning av alla valmöjligheter finns i boken i avsnittet om filhantering. Vi ska titta på ett första exempel där vi läser hela filinnehållet i en fil och skriver ut det på skärmen.

Öppna fil för läsning

```
<?php
$fil = fopen("filnamn.txt","r") or die("Kunde inte öppna fil");
$text = fread($fil,filesize("filnamn.txt"));
echo "<PRE>$text</PRE>";
fclose($fil);
?>
```

Den första raden öppnar filen. Den börjar med att beskriva vilken variabel som ska agera som filhandtaget. Filhandtaget använder vi för att komma åt filen. Det första argumentet till fopen innehåller filnamnet. Eftersom vi inte skrivit in någon sökväg, antar PHP att vi menar en fil i samma katalog som skriptet ligger. Det andra argumentet "r" talar om att vi vill öppna filen för läsning (read). Till sist har vi hängt på en konstruktion som ser lite konstig ut, men vad den gör är att helt enkelt skriva ut raden "Kunde inte öppna fil" om inte filen fanns där vi trodde att den skulle finnas.

På rad 2 i skriptet läser vi in allt innehåll från filen i variabeln $text. Detta görs med fread, som tar filhandtaget som första argument, och antalet tecken vi vill läsa in som andra argument. I detta fall vill vi läsa in hela filen, och då måste vi veta filstorleken, vilken vi får fram med funktionen filsize. På nästa rad skriver vi ut hela filen på skärmen. Vi använder HTML-märket <PRE> för att radbrytningar i filen också ska synas på webbsidan. Till sist stänger vi filen med hjälp av fclose.

Öppna fil för skrivning

Nästa sak vi ska titta på är hur man öppnar en fil för skrivning. Med den metoden vi går igenom kommer allt innehåll från filen att raderas om filen existerar sedan

tidigare. För att kunna skriva till en fil behöver vi en annan typ av filrättighet. Den får vi genom att skriva chmod ugo+w filnamn.txt vid kommandoprompten.

```
<?php
$fil = fopen("filnamn.txt","w") or die("Kunde inte öppna fil");
$text = "Hej hopp!";
fputs($fil, $text);
fclose($fil);
echo "Texten $text har skrivits till fil.";
?>
```

Fopen används på nästan samma sätt, den enda förändringen är att vi bytt ut r mot w (write). För att skriva till filen använder vi fputs (vilket står för file put string). Sedan är allt klart. Det finns fler filfunktioner beskrivna i boken, bland annat för att navigera mellan olika kataloger.

En specialfiness med fopen är att man lika gärna kan använda den för att öppna en webbsida på en annan server:

```
$fil = fopen("http://www.dn.se/","r") or die(Gick ej");
$text = fread($fil,filesize("filnamn.txt"));
fclose($fil);
?>
```

Sedan kan man söka efter speciell information inuti texten, t.ex. för att ta ut viss information som man har användning för.

Övningar filhantering

Övning 9.1

Gör ett program som tar den inmatade texten ur ett formulärs "textarea" och sparar den i en fil.

Övning 9.2

Gör ett program som i en textruta frågar efter ett filnamn på servern. Kontrollera sedan filnamnet så att det endast innehåller bokstäver, siffror och punkt. Om kontrollen ger OK, så öppna filen och skriv ut filinnehållet på skärmen.

Övning 9.3

Utveckla programmet i övning 1 till ett enkelt gästboksskript. Skapa en sida kallad "Lägg till till gästbok, där användaren får fylla i namn, e-post-adress och meddelande. När användaren skickar i väg formuläret ska informationen sparas snyggt formaterad i en fil. Snyggt formaterad innebär att du har mellanrum mellan namnet och e-post-adressen, och ny rad (
) innan du skriver meddelandet, och dessutom ny rad (
) efter själva meddelandet. Obs! Använd append (a) som filöppningsmetod, ej write, eftersom du då skriver över tidigare innehåll! Längst ned på varje sida ska en rubrik med texten "Skrivet i gästboken" samt filinnehållet visas.

Övning 9.4

Skapa ett loggningsskript som sparar en filrad om varje besök i en fil kallad "log.txt". Loggningsskriptet ska logga användarens IP-adress, och vilken sida användaren kom ifrån, och vilken webbläsare användaren hade. Variabler skickas till PHP med denna information. Skapa ett nytt PHP-skript, och skriv

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

så hittar du denna information längst ned på sidan.

Övning 9.5

Gör ett PHP-skript som kort presenterar den nuvarande kursen på Ericsson-aktien. Ta kontakt med t.ex. http://finans.yahoo.se för att ta reda på kursen.

Sessioner

Lagra variabelvärden mellan sidor

Ett problem som ofta uppkommer när man programmerar i PHP är att variabler inte behåller sitt värde mellan olika webbsidor. Detta beror på att HTTP är ett s.k. lägeslöst protokoll.

Det finns olika metoder att spara variabelvärden mellan sidor. Ett sätt är att skicka med gömda formulärfält - men detta blir snabbt ohanterligt då variablerna är många. Ett annat sätt är att skicka med information i själva adressen. Då stöter man dock på säkerhetsproblem. Dessutom finns en gräns i att en webbadress inte får vara längre än c:a 250 tecken.

För att göra det hela enklare har PHP stöd för sessioner. Detta innebär att man på ett enkelt sätt kan ha tillgång till samma variabelflora på hela webbplatsen, vilket är mycket använd bart för att t.ex. kontrollera att en användare är inloggad, eller för att genomföra ett köp i en webb-butik.

Varje session har ett unikt id

När en session startas i PHP får den ett unikt id. Denna långa sträng av tecken är det enda som skickas fram och tillbaka mellan besökarens webbläsare och webbservern för att identifiera vilken session en viss användare tillhör. Detta sker vanligtvis med hjälp av en kaka - en liten textfil som sparas i webbläsarens minne. Om besökaren stängt av kakfunktionen känner PHP av det och lägger i stället till sessionsid:t i slutet av webbadressen.

Använda en session

För att starta en session använder man sig av funktionen session_start. Den måste komma allra högst upp i PHP-skriptet, annars får man felmeddelandet Headers already sent. Därefter kan man registrera sessionsvariabler genom att använda matrisen $_SESSION:

fil1.php

```
<?php
session_start(); // Startar sessionen
$_SESSION['namn'] = "test";
?>
<A HREF="fil2.php">Fil 2</A>
```

Använda sessionsvariabler

Varje php-sida som använder sessioner ska ha ett anrop till session_start högst upp i skriptet. När session_start har anropats så hämtas ala sessionsvariabler. Dessa finns sedan i matrisen $_SESSION.

fil2.php

```
<?php
session_start();
echo $_SESSION['namn']; // Skriver ut test
?>
<A HREF="fil3.php">Fil 3</A>
```

Avsluta en session

Genom att tömma matrisen $_SESSION tas alla sessionsvariabler bort. För att helt avsluta sessionen anropas sedan session_destroy.

fil3.php

```
<?php
session_start();
$_SESSION = Null; // Tar bort alla sessionsvariabler
session_destroy() // Avslutar sessionen
echo $_SESSION['namn']; // Skriver inte ut någonting.
?>
```

Övningar sessioner

Övning 10.1

Skapa en enkel räknare som visar hur många gånger man varit inne på sidan. Spara sidan med namnet raknare.php och gör sedan en "submit"-knapp som går til l samma sida (raknare.php). Kontrollera att räknaren räknar upp 1 steg för varje gång man varit inne på sidan.

Övning 10.2

Utveckla programmet i övning 1 med en kryssruta där man kan bocka för att räknaren ska nollställas. Om kryssrutan är ikryssad när formuläret skickas iväg ska räknaren börja om från 0.

Övning 10.3

Skapa ett inloggningssystem där användaren kan logga in på en sida via ett formulär och när användarnamn och lösenord är kontrollerade ska användaren kunna gå runt på tre-fyra olika sidor. Användaren ska också kunna logga ut, och därefter inte kunna komma åt sidorna.

Databasteori

Det finns några bra skäl till varför man ska använda databaser för att lagra information. Bland annat är det mycket lättare att sortera ut den information man vill ha ur en databas. Detta görs med SQL som vi kommer att titta på senare idag. Men först lite allmän teori kring databaser.

Tabeller

En databas innehåller minst en men oftast flera tabeller. Dessa tabeller är uppbyggda på samma sätt som tabellerna i ett kalkylprogram, som t.ex. Excel. Varje tabell består av ett antal kolumner, som definieras när tabellen skapas. För en kundtabell kan vi t.ex. specificera kolumnerna namn, adress, postadress, telefon och kundnr. När man specificerar kolumnerna talar man också om för databashanteraren vad för typ av data kolumnen innehåller. För kundnumret använder man t.ex. datatypen INT (heltal) och för de andra kolumnerna antagligen VARCHAR som betyder sträng av valfri längd.

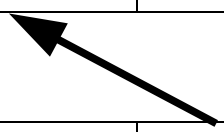
En av kolumnerna, eller attributen som kolumnerna också kallas, måste vara nyckel för tabellen. Varje värde i nyckelkolumnen måste vara unkt för en viss rad, d.v.s. det får inte förekomma två rader med samma värde i nyckelkolumnen. Varje gång man fyller på kundtabellen med en ny kund skapas en ny rad i tabellen.

Scheman och relationer

Innan man börjar med det rent praktiska arbetet med att skapa databastabellerna, bör man skissa på ett schema för databasen. Detta kan göras med hjälp av ett E-R-diagram, eller genom att helt enkelt skissa upp de tabeller som ska ingå i databasen, och dra pilar mellan tabellerna där kopplingar finns.

kunder

KundID	Fornamn	Efternamn	Adress	Postnr	Postort
1	Anders	Andreasson	Storgatan 3	10521	Stockholm
2	Berit	Carlsson	Kungsgatan 14	41105	Göteborg



order

OrderID	KundID	Summa	Datum
1	2	250	2005-12-01
2	1	350	2005-12-01
3	2	420	2005-12-15

Relationer mellan tabeller

Som ni ser finns det alltid kopplingar eller relationer mellan de två tabellerna. Relationen består här i att CustomerID i tabellen Orders refererar till CustomerID i tabellen Customers. Det finns tre olika typer av relationer: en till en (1:1), en till många (1:N) samt många till många (N:M). Vi har t.ex. en 1:1 relation mellan en kund och en kunds adress. 1:N relation har vi mellan en kund och kundens order, eftersom en kund kan lägga flera order.

Många till många relationer ordnar man genom att skapa en mellanliggande tabell. Säg till exempel att vi tabell över böcker och en över författare. Så länge varje bok endast har en författare får vi en 1:N-relation. Men i många fall har vi ju flera författare till en bok. Då har vi stället en N:M relation, och behöver skapa en extra tabell.

Några tips för design av databaser

- Tänk objektorienterat.
Med detta menas att du oftast behöver en tabell för varje objekt, t.ex. en tabell för kunder en för produkterna etc.

- Lagra inte samma data på två ställen.
Ska du t.ex. skapa ett ordersystem så lägg inte adressinformationen i både order- och kundtabellen. Detta innebär att samma information finns på två ställen, dels i ordertabellen, dels i kundtabellen. Om man nu ändrar kundens adress på ett ställe men inte på det andra, får man inkonsisten information - man vet inte vilken adress som är riktig av de två. Det är viktigt att tänka efter noga när man designar sin databas, det är mycket, mycket svårare att göra ändringar när väl alla data är inmatade och programmet för att hantera databasen skrivet!

- Ett värde i varje kolumn.
Om du har en tabell över order, ska du inte sätta alla varor som beställts i en kolumn. Skapa i stället en ny tabell (orderinnehåll) där du listar vad som beställts. Denna process kallas normalisering, och finns detaljerat beskriven i mer teoretiska databasböcker, som t.ex. Elmasri: Fundamentals of Database Systems.

- Tänk på frågorna.
När du designar databasen ska du ha en lista med helst alla frågor som det är aktuellt att ställa till databasen. Kontrollera att det är möjligt att få svar på frågorna med den aktuella designen.

- Få tomma värden.
Om vi t.ex. antar att vi utvecklar ett system för en internetbokhandel där kunderna ska kunna lägga in rescensioner, så kan det vid första anblicken verka logiskt att lägga in rescensionerna i boktabellen. Men eftersom denna tabellkolumn kommer att vara tom i de flesta fall är det bättre att bryta ut den kolumnen, och lägga i en separat tabell tillsamans med bokid.

SQL

Databashanterare

För att använda frågespråket SQL (Standard Query Language) behövs en databashanterare. De två vanligaste databashanterarna som används med PHP är MySQL och PostgreSQL. Instruktionerna i det här avsnittet är anpassade för båda servertyperna.

Att komma åt MySQL

Det första vi gör är att logga in i MySQL. Det gör vi med följande kommando:

```
mysql -h 127.0.0.1 -u användarnamn -D databasnamn -p
```

vid kommandoprompten. Adressen till den server som vi vill ansluta till står efter -h, 127.0.0.1 betyder den egna datorn. Efter -u skriver vi det användarnamn som har konfigurerats, och efter -D namnet på databasen. Har vi inte konfigurerat någon databas ännu kan -D utelämnas. Genom att ange -p anges att ett lösenord kommer att skrivas in på nästa rad.

Att komma åt PostgreSQL

Motsvarande kommando för PostgreSQL är psql. Det används på följande sätt:

```
psql -h 127.0.0.1 -d databasnamn -U användarnamn
```

Adressen till den server som vi vill ansluta till står efter -h, 127.0.0.1 betyder den egna datorn. Efter -d skriver vi namnet på databasen vi vill ansluta till, och efter -U användarnamnet. På nästa rad kommer vi att få förfrågan om lösenordet för användaren.

Skapa tabeller och fylla dem

Jag tänker nu använda en exempeldatabas för att demonstrera de vanligaste kommandona i SQL. Vi börjar, naturligt nog, att skapa databasen vi sedan kommer att lägga tabellerna i. Detta gör vi såhär:

```
CREATE DATABASE testdatabas;
```

Vi måste sedan tala om att vi vill använda just den databasen, detta gör vi i MySQL med kommandot USE. (Detta kommando kan variera mellan olika databashanterare)

```
USE testdatabas;
```

I PostgreSQL heter i stället kommandot connect och skrivs på kortast sätt så här:

```
\c testdatabas;
```

Nästa steg är att skapa tabellerna vi ska använda. Först en beskrivning av hur det hela ska se ut:

employee

employee_id	**name**	**telephone**
2	Arne	463212
3	Kalle	261456
4	Berit	173190
1	Cecilia	260032
5	David	210023

works_for

employee_id	**departmentID**
1	A
1	B
1	C
2	C
3	A
4	B

department

department_id	**name**	**boss**
A	Marknadsföring	3
C	Personal	2
B	Kundsupport	4

Tabellerna beskriver ett fantasiföretag som har fem anställda och hela tre avdelningar. Den första tabellen är ganska enkel att förstå; det är helt enkelt en lista over de anställda med deras anställningsnummer, namn och telefonnummer.

I mitten har vi en tabell som beskriver vilka som arbetar på vilka avdelningar. Som ni ser så arbetar Cecilia en hel del övertid, eftersom hon arbetar på tre avdelningar samtidigt!

Notera också att David är inte tillhör någon avdelning för närvarande, eftersom han är barnledig för tillfället. Den sista tabellen beskriver avdelningarna, med en bokstav för varje avdelning, avdelningens namn samt anställningsnumret på den person som är chef för avdelningen.

Att skapa tabellerna och fylla dem med innehåll är två separata steg. Vi börjar med att skapa tabellerna. För detta använder man kommandot CREATE TABLE;

```
CREATE TABLE employee (
employee_id int not null,
name varchar(20),
telephone varchar(10),
primary key (employee_id)
);
```

Precis efter CREATE TABLE kommer namnet på tabellen vi vill skapa. Det är praktiskt att hålla sig till engelska bokstäver för att vara på den säkra sidan - svenska tecken kan potentiellt ge problem p.g.a skiftande teckentabeller. Varje rad (utom den sista) beskriver sedan varsin kolumn i tabellen.

Först kommer namnet på kolumnen, därefter vad för typ av data kolumnen ska rymma och till sist kan vi ha lite extra inställningar längst till höger. Den första raden är av typen int, vilket står för integer och betyder heltal. Den får inte vara tom, och därför har vi lagt dit den extra inställningen "not null". Observera att null stavas med u.

De två nästa raderna är av typen varchar, det vill säga strängar med flexibel längd. Längden kan vara upp till 20 tecken för namnet och 10 tecken för telefonnumret. Därefter specificerar vi att employee_id ska vara nyckel för tabellen. Detta innebär att varje anställd måste ha ett unikt idnummer, på samma sätt som alla svenskar har varsitt unikt personnummer. En nyckel är nödvändig för varje tabell, eftersom man med dess hjälp alltid är säker på att få ut exakt en rad.

```
CREATE TABLE works_for (
employee_id integer not null,
department_id char(1) not null,
primary key (employee_id, department_id)
);

CREATE TABLE department (
department_id char(1) not null,
name varchar(20),
boss integer,
primary key (department_id)
);
```

Nästa två tabeller vi skapar är works_for och department. Notera att vi för den sista tabellen valt att ha två kolumner tillsammans som nyckel. Ett alternativ till detta hade varit att infoga en extra rad med ett unikt id som nyckel.

Till sist ska vi nu fylla tabellerna med data. Det gör vi med följande kommandon:

```
INSERT INTO employee (employee_id, name, telephone)
VALUES (2, 'Arne', '463212');

INSERT INTO employee (employee_id, name, telephone)
VALUES (3, 'Kalle', '261456');

INSERT INTO employee (employee_id, name, telephone)
VALUES (4, 'Berit', '173190');

INSERT INTO employee (employee_id, name, telephone)
VALUES (1, 'Cecilia', '260032');

INSERT INTO employee (employee_id, name, telephone)
VALUES (5, 'David', '210023');
```

```
INSERT INTO works_for (employee_id, department_id)
VALUES (1, 'A');

INSERT INTO works_for (employee_id, department_id)
VALUES (1, 'B');

INSERT INTO works_for (employee_id, department_id)
VALUES (1, 'C');

INSERT INTO works_for (employee_id, department_id)
VALUES (2, 'C');

INSERT INTO works_for (employee_id, department_id)
VALUES (3, 'A');

INSERT INTO works_for (employee_id, department_id)
VALUES (4, 'B');

INSERT INTO department (department_id, name, boss)
VALUES ('A', 'Marknadsföring', 3);

INSERT INTO department (department_id, name, boss)
VALUES ('C', 'Personal', 2);

INSERT INTO department (department_id, name, boss)
VALUES ('B', 'Kundsupport', 4);
```

Som ni ser är syntaxen ganska enkel, vi skriver INSERT INTO följt av tabellnamnet. Därefter listar vi de kolumner som vi ska fylla i, och värdena som kolumnerna ska fyllas med.

Söka i en tabell

Nu ska vi titta på det kommando som man använder oftast i SQL, nämligen SELECT. Med SELECT kan man sortera ut de rader ur en ell er flera tabeller man vill ha fram. Vi börjar med den enklaste varianten.

```
SELECT * FROM employee;
```

Som ni ser får vi fram en lista över alla de som är anställda på företaget, med all info ur tabellen. Säg att vi bara vill ha med namnen på de som är anställda på företaget. Då kan vi specificera frågan mer noggrannt:

```
SELECT name FROM employee;
```

Detta betyder helt enkelt - direktöversatt - Välj ut kolumnen name ur tabellen employee. Som ni ser är SQL även vid första anblicken ganska lätt att förstå . Vi kan också välja att visa fler rader:

```
SELECT name,telephone FROM employee;
```

Och om vi nu bara vill ta reda på mer om Cecilia och inte är intresserad av de andra på företaget? Då får vi lägga till en WHERE-sats för att ta ut en viss delmängd ur tabellen. Observera att vi använder enkla citationstecken runt namnet, inte dubbla.

```
SELECT * FROM employee WHERE name='Cecilia';
```

Här får vi då ut all information om Cecilia. Det finns fler tecken än = man kan använda, de vanligaste finns listade i "PHP programmering" på sidan 234.

Använda flera villkor

Genom AND kan man kräva att flera villkor ska vara uppfyllda. OR används för att specificera flera alternativ där minst ett behöver vara uppfyllt.

```
SELECT * FROM department WHERE boss = '4' OR boss = '2';
```

Detta ger en lista på de avdelningar som leds av anställd nummer 4 eller nummer 2.

Matcha datum

Genom

```
SELECT * FROM databas
WHERE datum > '1999-12-31' AND datum < '2000-01-07';
```

Här får vi ut de datum som är efter den 31 december 1999 men före 7 januari 2000.

Matchning av mönster

Operatorn LIKE söker efter poster som börjar eller slutar på ett visst sätt.

```
SELECT * FROM employee
WHERE telephone LIKE '26%';
```

Detta ger en lista med alla som har telefonnummer som börjar på 26.

Söka i flera tabeller

En av de främsta anledningarna till användning av en databas, är den mycket stora flexibiliteten i utformningen av sökningarna. I databasteorin kallas en sökning i flera tabeller för join. Det finns flera typer av join, vi börjar med "full join". Den används för att lägga ihop innehållet i flera tabeller och fungerar ungefär som multiplikation. Med detta menas att alla rader i de två tabellerna kombineras med varandra. Har vi 4 rader i den första tabellen och 3 i den andra, blir alltså antalet resulterande rader 4*3 = 12.

```
SELECT * FROM employee,department;
```

Som ni ser blir resultatet ganska oanvändbart. Så därför tillfogar vi ett villkor till vår sökfråga för att få ut det vi är intresserade:

```
SELECT * FROM department,employee WHERE boss = employee_id;
```

Nu har vi fått en betydligt prydligare lista med enbart information om avdelningen och dess chef. VIll vi göra själva sökfrågan tydligare kan vi också tala om vilken tabell kolumnen boss respektive employee_id kommer från. Detta är nödvändigt om vi har två kolumner i olika tabeller som heter samma sak, men är bra annars också, eftersom det gör det lättare att tyda mer komplicerade frågor. Med denna notation ser frågan ut så här:

```
SELECT * FROM department,employee
WHERE department.boss = employee.employee_id;
```

Säg nu att vi vill ta reda på endast namnet på marknadsavdelningens chef. Frågan lyder:

```
SELECT employee.name
FROM department,employee
WHERE department.boss = employee_id
AND department.name = 'Marknadsföring';
```

Och snabbt som ett trollslag har vi fått reda på att det är Berit som leder företagets marknadsföringsavdelning.

Till sist ska vi använda alla tre tabellerna för att ta reda på namnen på alla som arbetar på avdelningen Personal. Den här frågan kan vara lite svårare att förstå:

```
SELECT employee.name
FROM employee,works_for,department
WHERE department.name = 'Personal'
AND department.department_id = works_for.department_id
AND works_for.employee_id = employee.employee_id;
```

Vi börjar med första raden där vi talar om att vi är intresserade på att få reda på namnet på personer ur tabellen "employee" dvs anställda. Nästa steg är WHERE-satsen som specificerar att vi ska använda alla tre tabellerna i vår databas. Till sist kommer WHERE satserna. Den första är enkel att förstå; det är personalavdelningen vi är intresserade av.

Med nästa rad så begränsar vi urvalet i works_for tabellen till endast de rader som gäller personalavdelningen.

Med sista raden länkar vi ihop works_for tabellen med employee tabellen. Hade vi endast velat ha den anställdas id hade inte detta behövts, eftersom det redan finns i works_for tabellen. Men för att få reda på namnet på personen får vi alltså gå detta extra steg.

Tänk på de två sista WHERE-satserna som länkar mellan de olika tabellerna, så blir frågan lättare att förstå.

Hitta rader som inte matchar

Vi har nu studerat den vanligaste JOIN-typen nämligen CROSS JOIN, som vanligtvis skrivs ut som ett komma (","). Som vi såg så matchar CROSS JOIN alla rader i den ena raden med alla i de andra raden *där ett visst villkor gäller*. Om vi då vill ha fram raderna där villkoret inte stämmer? Då använder vi den andra vanliga typen av JOIN - nämligen LEFT JOIN. Som vi märker i resultatet fylls då kolumnen department_id ut med NULL för t.ex. David, eftersom han inte jobbar på någon av avdelningarna.

```
SELECT employee.name, works_for.department_id
FROM employee LEFT JOIN works_for
ON employee.employee_id = works_for.employee_id;
```

Genom att använda specialoperatorn IS NULL kan vi få ut endast de rader med anställda som inte tillhör någon avdelning:

```
SELECT employee.name, works_for.department_id
FROM employee LEFT JOIN works_for
ON employee.employee_id = works_for.employee_id
WHERE works_for.department_id IS NULL;
```

Sortera rader

Med SQL har du möjlighet att välja i vilken sorteringsordning du vill ha ut ditt svar. Detta kan vara användbart t.ex. när du vill ha en alfabetisk lista.

```
SELECT name,telephone FROM employee ORDER BY name;
```

Som ni ser får vi ut en prydlig lista i namnordning. Det går även att sortera i baklänges ordning:

```
SELECT name,telephone FROM employee ORDER BY name DESC;
```

Visa ett visst antal rader

Med LIMIT kan man välja hur många rader av svaret man vill ha ut. Detta exempel tar ut de anställda vars namn kommer först i alfabetet:

```
SELET * FROM employee ORDER BY name LIMIT 2;
```

Detta är användbart om man har väldigt många poster i en tabell och vill visa några åt gången på en webbsida. För att visa nästa namn i tabellen efter de två första gör vi så här:

```
SELECT * FROM employee ORDER BY name LIMIT 2,1;
```

Det går också att "röra om" i ordningen så att den blir helt slumpmässig. Kanske lämpligt för att se vem som vinner i företagets konstlotteri?

```
SELECT * FROM employee ORDER BY RAND() limit 1;
```

Räkna rader

COUNT kan räkna hur många rader det finns som matchar ett visst sökbegrepp.

```
SELET COUNT(*)
FROM works_for,department
WHERE works_for.department_id = department.department_id AND
department.name = "Marknadsföring";
```

Nu har vi tagit reda på hur många som arbetar på marknadsföringsavdelningen.

Ändra rader

Vi har just fått reda på att Cecilia har fått nytt nummer, och behöver ändra det i tabellen. Detta gör vi med UPDATE-kommandot.

```
UPDATE employee SET telephone='363471' WHERE name='Cecilia';
```

Det är viktigt att ha med en WHERE-sats här, annars kommer allas telefonnummer att ändras till 363471!

Lägga till kolumner till en tabell

Säg att det lilla företaget har tagit steget in i IT-åldern och skaffat varje anställd en e-post adress. Hur gör vi för att lägga till en kolumn för att få med denna information? Vi använder då ALTER TABLE. Det går att göra många ändringar med detta kommando, alla möjligheter finns listade i boken "PHP Programmering" på sidan 225.

```
ALTER TABLE employee ADD COLUMN email VARCHAR(40);
```

Vi har nu lagt till kolumnen email, och begränsat längden på e-post-adressen till 40 tecken.

Ta bort data från tabell

Till sist ska vi ta bort en anställd från tabellen, eftersom han slutat på företaget. Detta görs med DELETE-kommandot. Var mycket försiktig med detta kommando! Om du råkar glömma WHERE-satsen tas nämligen hela tabellinnehållet bort!

```
DELETE FROM employee WHERE namn='Arne';
```

Ta bort en hel databas

Med hjälp av kommandot DROP DATABASE kan en hel databas, och alla tabeller i den, tas bort. Använd också detta kommando med stor försiktighet, eftersom inget varningsmeddelande ges. Det gäller alltså att kontrollera att man tar bort rätt databas!

```
DROP DATABASE databasnamn;
```

Övningar SQL

Övning 11.1

Skapa dessa tabeller. Avgör vilken av kolumnerna som passar som nycklar. Tänk på att använda datatypen DATE som passar för datum. Innan du matar in tabellerna så fundera också på om kolumnindelningen i tabellen kunder är optimal.

hyrbilar

märke	**namn**	**registreringsnr**
Ford	Escort	DLE475
Volvo	850	ASQ321
Saab	9000	DRE729
Volkswagen	Jetta	UTY413

kunder

namn	**adress**	**postadress**	**telefon**	**id**
Arne Nilsson	Brovägen 12B	414 32 Göteborg	031123421	3
Gunnar Carlsson	Hålltorp pl 42	421 12 Mölndal	031452123	1
Berit Radewski	Vilovägen 8	413 45 Göteborg	031641657	2

uthyrdabilar

registreringsnr	**kundid**	**startdatum**	**slutdatum**	**id**
DLF475	3	20010601	20010602	1
DRE729	2	20010602	20010602	2
DLF475	3	20010611	20010612	3
ASQ321	3	20010610	20010613	4

Övning 11.2

Arne har bytt telefonnummer. Det nya numret är 031241017. Lägg in denna ändring i databasen.

Övning 11.3

a) Ta fram en lista med komplett information om alla kunder.
b) Ta fram en lista med alla kunders namn.
c) Ta fram en lista med kundernas namn och adress till ett kundutskick. Listan ska vara sorterad efter postnummer eftersom företaget då får rabatt av Posten på försändelserna.
d) Ta fram en lista på alla kunder som någon gång hyrt en bil.
e) Kontrollera vilka bilmodeller som Arne Nilsson hyrt.
f) Kontrollera vilka bilar (komplett med märke, namn och registreringsnummer) som var uthyrda den 12 juni.
g) Ta fram en lista på kunder som har hyrt Forden.
h) Ta fram en lista på hur många gånger Forden blivit uthyrd.
i) Ta fram en lista på de bilar som inte blivit uthyrda.

Övning 11.4

Gunnar Carlsson ska tas bort från kunddatabasen. Ordna detta.

Övning 11.5

Biluthyrningsföretaget har kommit på att det kan vara bra att ha kundernas mobilnummer. Lägg därför till en extra kolumn för detta. Lägg också till mobilnumret för Berit som är 07014521344.

Övning 11.6

Biluthyrningsföretaget har beslutat att inrätta ett enkelt kvalitetssystem, där kunderna får sätta betyg mellan 1 och 5 på bilarna de hyrt. Inrätta en tabell för detta, och fundera på vilka fält den ska innehålla.

Övning 11.7

Du ska skapa en databas för en idrottsklubb. Databasen ska innehålla information om deltagare (med namn, adress, postnr, telefon) etc. Dessutom ska information finnas om vem som deltar i vilka aktiviteter. Aktiviteterna som finns är bowling, innebandy och volleyboll. Skissa på ett lämpligt databasdiagram för detta.

MySQL och PHP

Introduktion

För att komma åt databaser från PHP finns det ett antal fördefinierade funktioner. För MySQL finns funktionerna i serien mysql_... Det kan vara bra att använda ett s.k. abstraktionslager, där vi via ett antal generella funktioner kommer åt olika databaser. Då blir det mycket lättare att byta databas, eftersom man bara behöver ändra inställningarna på ett ställe. Ett exempel på ett funktionsbibliotek som innehåller generella databasfunktioner är PEAR som ingår i PHP.

En databasfråga från PHP

Vi kommer nu att använda samma exempeldatabas som förut och visa hur vi använder PHP för att komma åt den.

```
$db = mysql_connect("localhost", "root") or die("Gick ej");
mysql_select_db("testdatabas");
$query = "select * from employee";
$result = mysql_query($query) or die("Fråga gick ej");
echo "<HTML><HEAD><TITLE></TITLE></HEAD>";
echo "<TABLE BORDER>";
echo "<TR>";
echo "<TH>employee_id<TH>name<TH>telephone<TH>email</TR>";
while ($row = mysql_fetch_array($result)) {
      echo "<TR>";
      echo "<TD>" . $row["employee_id"];
      echo "<TD>" . $row["name"];
      echo "<TD>" . $row["telephone"];
      echo "<TD>" . $row["email"];
}
echo "</TABLE></BODY></HTML>";
```

Den första raden i skriptet ansluter till databasen.

På nästa rad väljer vi vilken databas vi ska använda. Detta motsvarar kommandot USE testdatabas om vi hade kommit åt MySQL direkt från kommandoprompten. Sedan sparar vi vår fråga i variabeln $query. Därefter skickar vi frågan till mysql med hjälp av mysql_query.

Inuti while-satsen har vi en konstruktion där vi hämtar en rad i taget från tabellen, och stoppar in värdena i matrisen row. Matrisen får automatiskt kolumnrubrikerna som index. Sedan kommer vi åt värdena i tabellen med hjälp av $row, och skriver ut dem snyggt formaterade i en tabell.

Man använder alltid mysql_query för att interagera med MySQL. Vill vi t.ex. lägga till en rad skickar vi följande i mysql_query:

```
INSERT INTO employee (employee_id, name, telephone, email)
VALUES ('5','Anders Ek','031475323','anders@ek.com')
```

Alla MySQL-kommandon i PHP finns beskrivna i manualen på adressen
http://www.php.net/manual/en/ref.mysql.php

Övningar MySQL och PHP

Övning 12.1

Skapa en tabell som innehåller CD-skivor, med uppgift om artistens namn och CD-skivans titel. Skapa sedan ett webbgränssnitt för att lägga till information till tabellen. Lösenord ska krävas för att få göra detta.

Övning 12.2

Använd samma tabell som i övning 12.1 och skapa en sida där det ska gå att söka i tabellen efter artister eller titlar. För detta ska ej lösenord behövas.

Övning 12.3

Modifiera skriptet och tabellen i övning 12.2 så att användare kan rösta på favoritartister. Användare ska kunna välja bland existerande artister och trycka på en knapp märkt "Rösta på denna artist". På en speciell röstsida ska man kunna se vilka som ligger tio-i-topp och hur många röster de har.

Övning 12.4

Skapa en adressbok, där användare ska kunna söka på namn, adress och epostadress. Användaren ska även kunna välja i en drop-down-meny om han/hon vill söka på alla företag eller välja mellan de företag som har personer med i adressboken.

PostgreSQL och PHP

Introduktion

För att komma åt databaser från PHP finns det ett antal fördefinierade funktioner. För PostgreSQL har funktionerna prefixet pg_ . PostgreSQL-funktionerna finns dokumenterade i PHP-manualen på http://www.php.net/manual/en/ref.pgsql.php .

Det kan vara bra att använda ett s.k. abstraktionslager, där vi via ett antal generella funktioner kommer åt olika databaser. Då blir det mycket lättare att byta databas, eftersom man bara behöver ändra inställningarna på ett ställe. Ett exempel på ett funktionsbibliotek som innehåller generella databasfunktioner är PEAR som ingår i PHP. Paketet DB som används för detta finns på http://pear.php.net/package/DB .

Visa en databastabell på en webbsida

Vi kommer nu att använda samma exempeldatabas som förut och visa hur vi använder PHP för att komma åt den. Fältet ADRESS ersätts med IP-numret på datorn där databasen finns. DATABAS ersätts med databasnamnet och ANVÄNDARE och LÖSEN med användarnamn respektive lösenord för att kunna ansluta.

```
$db = pg_connect("host=ADRESS dbname=DATABAS user=ANVÄNDARE password=LÖSEN") or
die('Kunde inte ansluta: ' . pg_last_error());
$query = "SELECT * FROM employee";
$result = pg_query($query) or die('Frågan misslyckades: ' . pg_last_error());
echo "<HTML><HEAD><TITLE></TITLE></HEAD>";
echo "<TABLE>";
echo "<TR>";
echo "<TH>employee_id</TH> <TH>name</TH> <TH>telephone</TH> <TH>email</TH>";
echo "</TR>";
while ($row = pg_fetch_array($result)) {
      echo "<TR>";
      echo "<TD>" . $row['employee_id'] . "</TD>";
      echo "<TD>" . $row['name'] ."</TD>";
      echo "<TD>" . $row['telephone'] . "</TD>";
      echo "<TD>" . $row['email'] . "</TD>";
      echo "</TR>";
}
echo "</TABLE>";
echo "</BODY></HTML>";
pg_free_result($result);
pg_close($db);
?>
```

På den första raden i skriptet ansluter vi till databasen med pg_connect. Sedan sparar vi vår fråga i variabeln $query. Därefter skickas frågan till PostgreSQL med pg_query.

Därefter inleds en tabell med märket <TABLE> och därefter kommer en tabellrad <TR> i vilken kolumnrubrikerna sätts mellan märkena <TH> och </TH>.

Inuti while-satsen finns en konstruktion där en rad i taget hämtas från tabellen, och värdena lagras i matrisen row. Matrisen får automatiskt databastabellens kolumnrubriker som index. Sedan kommer vi åt värdena i tabellen med hjälp av $row, och skriver ut dem snyggt formaterade i en tabell.

Lägga till data till en databastabell via ett formulär

Det är alltid pg_query som används för att skicka SQL-frågor till PostgreSQL, oavsett vilken typ av SQL-fråga det gäller.

I detta exempel tar vi formulärdata inmatade av besökaren och lagrar dessa i en databastabell. Det är alltid viktigt att kontrollera formulärdata så att innehållet och längden håller sig inom godkända ramar. I detta exempelprogram görs en kontroll av längden och specialtecken kommenteras bort med **addslashes**.

Vill vi t.ex. lägga till en rad i tabellen register med fälten name och telephone kan php-programmet se ut så här:

```
<HTML>
<HEAD>
<TITLE>Lägga till anställd</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<?php
if (!$_POST['skickat']) {
      echo "<H1>Lägg till anställd</H1>";
      echo '<FORM ACTION="' . $PHP_SELF . '" METHOD="POST">'
      echo 'Namn: <INPUT TYPE="text" NAME="name">';
      echo '<BR>';
      echo 'Telefonnummer: <INPUT TYPE="text" NAME="telephone"> ';
      echo '<INPUT TYPE="hidden" NAME="skickat" VALUE="1">';
      echo '<BR><INPUT TYPE="submit" VALUE="Skicka">';
} else {
      $name = addslashes($_POST['name']);
      $telephone = addslashes($_POST['telephone']);
      $name = substr($name,0,100);
      $telephone = substr($telephone,0,50);
      $query = "INSERT INTO register (name, telephone)
                VALUES ('$name', '$telephone')";
      $db = pg_connect("host=ADRESS dbname=DATABAS user=ANVÄNDARE
            password=LÖSEN") or die('Kunde inte ansluta: ' . pg_last_error());
      $result = pg_query($query) or die('Frågan misslyckades: '  .
            pg_last_error());

      echo "Posten har lagts till";
}
?>
</BODY>
</HTML>
```

Övningar PostgreSQL och PHP

Övning 13.1

Skapa en tabell som innehåller CD-skivor, med uppgift om artistens namn och CDskivans titel. Skapa sedan ett webbgränssnitt för att lägga till information till tabellen. Lösenord ska krävas för att få göra detta. Skapa sedan ett till gränssnitt för att söka i tabellen efter CD-skivor eller titlar.

Övning 13.2

Använd samma tabell som i övning 13.1 och skapa en sida där det ska gå att söka i tabellen efter artister eller titlar. För detta ska ej lösenord behövas.

Övning 13.3

Modifiera skriptet och tabellen i övning 13.2 så att användare kan rösta på favoritartister. Användare ska kunna välja bland existerande artister och trycka på en knapp märkt "Rösta på denna artist". På en speciell röstsida ska man kunna se vilka som ligger tio-i-topp och hur många röster de har.

Övning 13.4

Skapa en adressbok, där användare ska kunna söka på namn, adress och epostadress. Användaren ska även kunna välja i en drop-down-meny om han/hon vill söka på alla företag eller välja mellan de företag som har personer med i adressboken.

Fri mjukvara

Vad är fri mjukvara & öppen källkod?

PHP i sig självt distribueras med öppen källkod. Detta innebär att vem som helst kan se hur PHP-språket är utformat. På engelska använder man ofta beteckningen "free software" - mjukvara som är gratis för alla och fri att använda. Detta särskiljer PHP från t.ex. ASP. ASP är ett språk utvecklat av Microsoft som används för liknande tillämpningar som PHP.

PHP ingår i en större flora av öppen mjukvara dit även Perl, Linux, Apache och ett flertal andra program tillhör. Denna kultur av fri kod innebär också att det i PHP finns en stor mängd av färdiga program och bibliotek att hämta. Detta innebär att man ofta slipper återuppfinna hjulet utan i stället kan bygga på andras kod och därmed förkorta utvecklingsprocessen kraftigt.

Adresser till PHP-skript och bibliotek

SourceForge är ett stort samlingsställe för projekt inom öppen mjukvara. Sourceforges PHP-avdelning hittar man på:
http://sourceforge.net/softwaremap/trove_list.php?form_cat=183

PHPMyAdmin är ett system för att hantera MySQL-databaser via webben.
http://www.phpmyadmin.net/

PHPPgAdmin är ett liknande system för att hantera PostgreSQL-databaser via webben.
http://www.phpmyadmin.net/

HotScripts har en stor samling PHP-skript sorterade efter kategori
http://www.hotscripts.com/PHP/

The PHP Resource Index har en något mindre samling PHP-skript
http://php.resourceindex.com/

Referenser

Böcker

Jonsson, Viktor (2001) *Webbprogrammering med PHP*, Lund: Studentlitteratur
Den bästa svenska boken om PHP.

Welling, Luke et.al. (2005) *PHP and MySQL web development*, Indiapolis: Sams Publishing
Bok för den som programmerat i ett annat språk innan och vill gå direkt på mer avancerade exempel.

Sklar David et.al. (2002) *PHP Cookbook*, Sebastopol: O´Reilly
Innehåller kodexempel för PHP.

Dubois, Paul (2005) *MySQL*, Indiapolis: Sams Publishing
Omfattande referens till MySQL - över 1000 sidor!

DuBois, Paul (2006) *MySQL Cookbook*, Sebastopol: O'Reilly
Innehåller många fiffiga kodrecept för MySQL.

Gilmore, Jason (2006) *Beginning PHP and PostgreSQL 8*, Berkeley: Apress
Lämplig för användare på medelnivå eller högre som vill lära sig PHP och PostgreSQL.

Douglas, Korry (2005) *PostgreSQL*, Indiapolis: Sams Publishing
För den som vill veta mer om PostgreSQL:s mer avancerade funktioner.

Friedl, Jeffrey (2002) *Mastering regular expressions*, Sebastopol: O´Relly
Fördjupning om reguljära uttryck / mönstermatchning.

Tidskrifter

PHP arch
Bästa tidskriften om PHP. Publiceras både som PDF och i tryckt form (för prenumeration).
http://www.phparch.com/

PHP Magazine
En tryckt tidskrift om PHP med en del gratisartiklar på webben.
http://www.php-mag.net/

Länkar

För länkar till skriptbibliotek se föregående avsnitt "Fri mjukvara".

PHP.net
Huvudsidan för PHP.
http://www.php.net/

PHP Portalen
Många svenska artiklar och kodexempel.
http://www.phpportalen.net/

PHPBuilder
Innehåller många artiklar med kodexempel och har ett populärt diskussionsforum.
http://www.phpbuilder.com/

OnLAMP PHP Devcenter
O´Reillys artiklar om PHP, bland annat med utdrag från deras böcker.
http://www.onlamp.com/php/

Zend Developer Zone
Många bra artiklar från gruppen bakom skriptmotorn Zend som är basen i PHP.
http://devzone.zend.com/

OpenDirectory PHP
Bra länkguide till sidor om PHP
http://www.dmoz.org/Computers/Programming/Languages/PHP/

PHP.net links
Fler länkar finns på PHP.net:s länksida.
http://www.php.net/links.php