Malmö högskola
Centrum för teknikstudier

# Webbprogrammering, DA123A, Studiemateriel 7

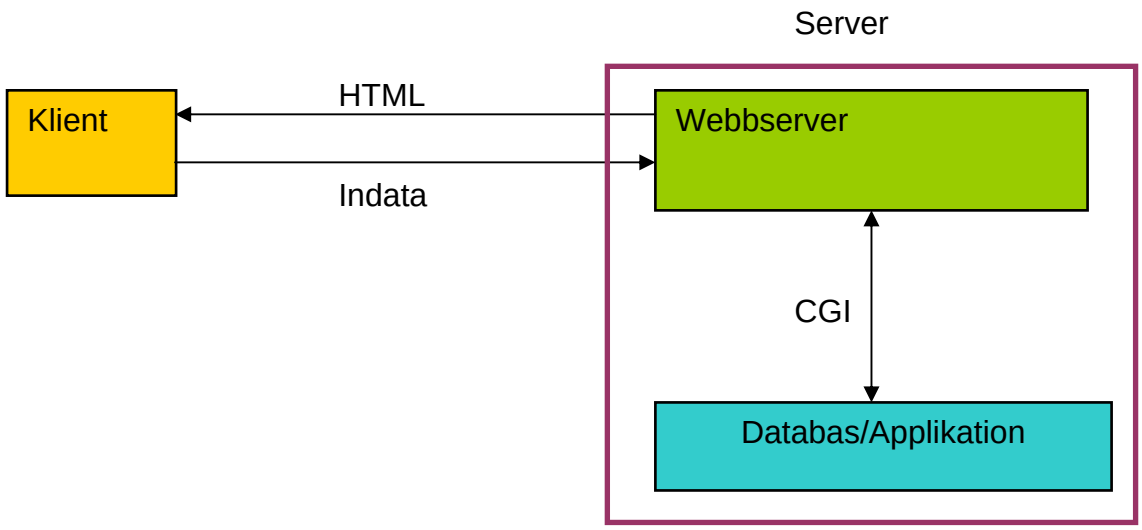
Detta materiel (och flera följande) innehåller en stor mängd adresser till olika webbsidor. Visa av dessa står direkt i texten och andra finns i fotnoter. Så vad är då skillnaden - varför står vissa i texten och vissa inte?

De adresser som står i texten förutsätts du som student titta på och läsa som en del av studiematerielet. De länkar som står i fotnoter finns för den som är intresserad och vill veta mer. Vissa adresser som står i fotnoter i ett materiel kan dyka upp i texten för andra.

## Introduktion CGI

### Dataflödet

CGI står för *Common Gateway Interface* och är skapat för att möjliggöra interaktion via webbservern och annan programvara, till exempel databaser. CGI är inget språk i sig utan är en specifikation för hur en webbserver ska kommunicera med ett skript/program. Detta ger möjlighet till dubbelriktad kommunikation. Användaren som besöker hemsidan kan ”köra program” på servern. CGI medför alltså att den som surfat till din webbplats kan få information från vissa externa program på din server, till exempel uppgifter från en databas. Som vi ska se senare så medför denna teknik många fördelar så som interaktivitet på hemsidan. Bilden nedan demonstrerar dataflödet med CGI.




HTML-sidan som klienten får är alltså dynamiskt genererad via en applikation och kan ha olika utseende beroende på indata som klienten har skickat. I ett PHP-script så händer följande när en webbläsare begär ett dokument med php-ändelse:

1. Klienten skickar en förfrågan om att få en viss sida med eventuell indata.
2. Webbservern ser ändelsen och sänder en förfrågan till PHP-tolken.
3. PHP-tolken scannar filen efter PHP-kod
4. När PHP-token hittar PHP-kod så byter den ut koden mot den utdata koden genererar. Till exempel hämtas data från en databas.
5. En HTML-fil med utdata skickas tillbaka till webbservern.
6. Webservern skickar vidare HTML-filen till webbläsaren.
7. Webbläsaren visar HTML-filen.

**Flexibilitet**

Det finns många olika programmeringsspråk som kan används tillsammans med CGI vilket gör det till en mycket flexibel teknik. Nedan visas ett exempel på hur en enkel loop kan se ut i några av de programmeringsspråk som finns med CGI-stöd.

JavaScript:

```
<html>
<div>
Javascript:<br>
<script type="text/javascript"
for (var i = 1; i <= 10; i++) {
  document.write("Hej, detta är rad nr "+i+"<br>");
}
</script>
</div>
</html>
```

Perl:

```
#!/usr/bin/perl -w
print "Content-type: text/html\n\n";
print "<html>Perl:<br>\n";

for ($i = 1; $i <= 10; $i++) {
    print"Hej, detta är rad nr ". $i ."<br>\n";
}
print "</html>\n";
```

ASP:

```
<html>
ASP:<br>
<%
for i = 1 to 10
  response.write("Hej, detta är rad nr "& i &"<br>")
next
%>
</html>
```

C:

```
// C-kod
#include <stdio.h>
main()
{
  int i;
  printf ("Content-type: text/html\n\n");
  printf ("<html>C:<br>\n");
  for (i = 1; i <= 10; i++) {
    printf ("Hej, detta är rad nr %d<br>\n", i);
  }
  printf ("</html>\n");
}
```

PHP:

```
#!/usr/bin/php
<html>PHP:<br>
<?
for ($i = 1; $i <= 10; $i++) {
    print "Hej, detta är rad nr ". $i ."<br>\n";
}
?>
</html>
```

Alla skripten ovan innehåller en loop som skriver ut samma rad 10 gånger vilket ger denna bild för användaren.

Hej, detta är rad nr 1
Hej, detta är rad nr 2
Hej, detta är rad nr 3
Hej, detta är rad nr 4
Hej, detta är rad nr 5
Hej, detta är rad nr 6
Hej, detta är rad nr 7
Hej, detta är rad nr 8
Hej, detta är rad nr 9
Hej, detta är rad nr 10

Observera hur vissa språk kan baka in sin kod mellan HTML-avsnitt, så som PHP, ASP och JavaScript medan vissa så som c och Perl måste generera all HTML via språket.

Tillämpningar av CGI kan vara

- Formulärhantering
- Dynamiska sidor
- Inloggning
- Klickbara bilder
- Besöksräknare
- Respons från kunder
- Databaskoppling

CGI är som sagt språkoberoende i det att om ett språk finns med ett gränssnitt mot CGI så går det att använda. Detta är inte alltid så självklart eftersom språket ifråga måste stöda den miljö (operativsystem) som webbservern arbetar under. Dessutom måste det vara installerat på webbservern ifråga. Tekniken är oberoende av klienten (under förutsättning att inga specialtaggar används).

**Fördelar och nackdelar**

Som alla tekniker så har CGI både fördelar och nackdelar att väga mot varandra när man väljer att använda dem eller inte.

Fördelar med CGI:

- CGI är inte beroende av ett visst språk vilket medför att det fungerar på många olika plattformar.
- Utbudet av olika språk gör att man inte behöver lära sig ett speciellt språk om man vill utveckla för CGI.
- Skapar dynamiska sidor - gör webbplatsen enklare att administrera och mer interaktiv för besökaren.

Nackdelar:

- Belastar servern. Användandet av dynamiska sidor via CGI belastar servern då en måste skapa HTML-sidor dynamiskt och eventuellt köra externa program/hämta data från en databas.
- Vid interaktiva sidor så måste hela sidan laddas om vid en förändring. Med exempelvis JavaScript på klientsidan så kan du via DOM ändra de delar av sidan som berörs. Med serverbaserade tekniker så måste data skickas till servern som genererar en ny sida som kanske bara har en liten förändring men hela den nya sidan måste laddas in i webbläsaren igen.
- Måste vara rätt konfigurerad annars kan det vara en säkerhetsrisk.

Fördelarna är uppenbara: dina besökare kan göra mer på din webbplats. Nackdelarna bör ni lägga märke till. Säkerhetsaspekten är den klart viktigaste. Om man vid till exempel mottagning av formulärdata inte gör någon indatakontroll, så kan det vara en säkerhetsrisk.
Dessutom är det så att det program som anropas körs på samma dator som webbservern, och belastar denna.

**Förbättringar gentemot CGI**
PHP & ASP kan till skillnad från vissa andra skriptspråk köras direkt i webbservern istället för att köras via CGI. Vilket ger en bättre prestanda då ett kommunikationslager försvinner. ASP och PHP är skriptspråk, det vill säga koden kompileras inte utan interpreteras (tolkas) under exekveringen.

Språken och plattformar utvecklas, detta gör att prestanda är direkt beroende på vilken hårdvara, webbserver-program och konfiguration man har. Vilket språk man skall välja för en lösning beror alltid på VAD man ska göra för något, men i grova drag kan vi säga att:

- ASP och PHP är likvärdigt både prestanda och funktionellt.
- PHP utvecklas snabbare än Perl och är idag mer populärt.
- Perl har fortfarande fler färdiga moduler än PHP/ASP och underlättar vid till exempel bildhantering etcetera.

**Fler språk**
För den intresserade så listas här några fler språkalternativ för CGI:

- JSP - Java server pages
- ASP - Active server pages (ny version heter ASP.NET)
- ISAPI, NSAPI (DLL[1]:er)
- SSI - Server Side Includes (en av föregångarna till PHP)
- Kompilerad C-fil som laddas som modul i webbserver
- Kompilerad C-fil som körs genom CGI

JSP är i HTML-koden inbäddad Java-kod som genererar en så kallad servlet som kompileras till färdig HTML av servern innan den skickas till klienten.

ASP
Ursprungligen en teknik från Microsoft, men numera finns fri programvara som möjliggör att ASP kan köras på varierande plattformar. ASP.NET ny version av ASP från Microsoft.

ISAPI och NSAPI
ISAPI (Internet Server Application Programme Interface) och NSAPI (Netscape Server API). Dessa tekniker har den fördelen att DLL:erna stannar kvar i minnet även efter det att programmet har avslutats (till skillnad från vanliga program). Det går alltså mycket snabbt att anropa program i DLL:erna andra gången programmet anropas.

**Ladda hem script**
Det finns en mängd färdiga script för CGI att ladda hem via Internet. Det gäller dock att tänka sig för när man gör detta:

- Vilka är kraven för ditt script?
- Skriv ut och titta på källkoden i detalj. Är den kommenterad? Går det att förstå vad som händer?
- Finns det några säkerhetsluckor som programmeraren inte har tagit hänsyn till?
- Testa, testa och testa igen!

Det viktigaste när du laddar hem ett script för att använda är att du förstår vad scriptet gör i sin helhet (var observant på Perl-script som kan vara svårlästa). Om du inte förstår scriptet så bör du inte använda det.

## Ajax
Ajax[2] står för Asyncronous JavaScript And XML och är en metod för att slippa ladda om en hel sida men ändå kunna hämta nytt materiel från en server. Idag har metoden blivit mer omfattande än när den först presenterades och fler element ingår i konceptet, exempelvis DOM och CSS.

Man använder JavaScript-klassen XMLHttpRequest (Microsoft gjorde detta som ett ActiveX-objekt i IE 5 och 6 men har sedan IE 7 samma interface som Firefox) för att hämta XML-data från servern (det går bra med andra dataformat också, exempelvis JSON[3]). Med hjälp av JavaScript och DOM (via HTML eller XML) så kan man sedan manipulera sidan utifrån den mottagen data. Användandet av XML-data öppnar för möjligheter tillsammans med XHTML.

[1] Dynamic Link Library
[2] http://en.wikipedia.org/wiki/AJAX_(programming)
[3] http://en.wikipedia.org/wiki/JSON

Med ordet asynkron avses att man kan ha flera förfrågningar i gång samtidigt och att ordningen på svaren på förfrågningarna till servern inte nödvändigtvis behöver komma i samma ordning som man initierade förfrågningarna.

Ajax är smidigare än CGI på så sett att man inte laddar om hela sidan och man kan göra dynamiska användargränssnitt som blir snabbare än med CGI.

Nackdelarna med Ajax ligger i att det krävs en del extralösningar för att få funktioner i webbläsaren som är knutna till sidans status att fungera som användaren kanske är van vid. Ajax öppnar också en säkerhetsrisk mot servern på samma sätt som CGI.

Det finns inte utrymme i den här kursen för att gå in närmare på Ajax men, framförallt för den som läst Webbutveckling fortsättningskurs och har CSS med sig, så har man efter den här kursen en hel del av de grunder som krävs för att knyta ihop teknikerna som utgör Ajax.

## PHP - introduktion

PHP står för *PHP Hypertext Preprocessor* och distribueras som fri programvara (med en licens[4] lik GNU GPL[5]), vilket innebär att alla som vill kan modifiera programmeringsspråket och sprida sin version vidare utan kostnad. Att PHP är fri programvara innebär också att alla som vill kan börja utveckla språket vidare och de bästa lösningarna inkorporeras i språket. Vad det innebär i praktiken är att PHP-språket utvecklas i en allt snabbare takt jämfört med till exempel ASP som har företag bakom sig med begränsningar i utvecklingsavdelningar etcetera.

PHP lämpar sig speciellt för webbutveckling och kan bäddas in i HTML. Dess syntax liknar C, Java samt Perl och är enkel att lära sig. Det huvudsakliga målet med språket är att göra det möjligt för webbutvecklare att skriva dynamiska webbsidor snabbt, men du kan förstås göra mycket mer med PHP. Komplett dokumentation och manual för PHP finns på **www.php.net**

En fördel med PHP är faktiskt det faktum att det är mycket väl dokumenterat och den som utvecklar webbsidor (eller annat) i PHP har ett fantastiskt stöd i den information som finns på php.net. Kurslitteraturen tar upp detta och ger dig en mängd användbara länkar som du definitivt bör bekanta dig med så därför går vi inte in närmare på detta här utan hänvisar till kurslitteraturen. Det finns en svensk spegel av php.net: http://se2.php.net/

### Användningsområden

PHP har flera användningsområden men det vanligaste är för att skapa dynamiska webbsidor enkelt och detta är också målet med PHP som programmeringsspråk. Nedan är en liten översikt över olika användningsområden.

- Webbsida
  - Huvudanvändningsområde
  - Skapar dynamiska webbsidor, kan ha kontakt med databaser, skickar e-post, modifierar bilder etc.

[4] http://se2.php.net/license/3_01.txt
[5] http://www.gnu.org/copyleft/gpl.html
http://en.wikipedia.org/wiki/GNU_General_Public_License
http://sv.wikipedia.org/wiki/GNU_GPL

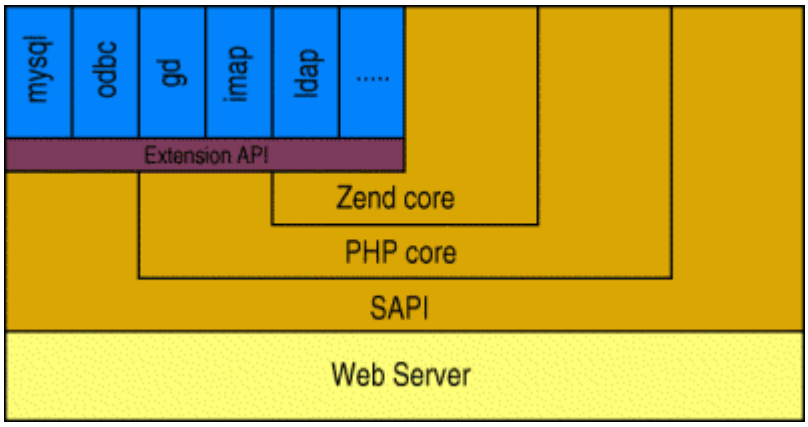
- Shellscript
  - Stora webbplatser, tidsstyrda skript
  - Klar fördel att kunna skriva dessa script i samma språk som webbsidorna.
- GUI - Grafiska applikationer GUI står för Graphical User Interface - Eller Grafiskt användargränssnitt på svenska jämför Windows med till exempel DOS. (Med grafisk gränssnitt styr man med t.ex. en mus till skillnad från ett textbaserat gränssnitt som man styr med tangentbord.)

**Databaser och PHP**
PHP är väl utvecklat för att interagera med databaser. Vi kommer att gå in närmare på hur PHP fungerar med databaser senare i kursen och ger därför bara en snabb överblick här.
Databaser som stöds av PHP:

– Adabas D
– Ingres
– Oracle(OCI7 and OCI8)
– dBase
– InterBase
– Ovrimos
– Empress
–
– FrontBase
– PostgreSQL
– FilePro (read-only)
– mSQL
– Solid
– Hyperwave
– Direct MS-SQL
– Sybase
– IBM DB2
– MySQL
– Velocis
– Informix
– ODBC
– Unix dbm

Det finns fler men bara denna lista ger ett intryck av att detta verkligen är något man siktat mot med PHP. ODBC[6] *Open DataBase Connectivity* är en standard för att ansluta till databaser. De flesta databaserna stöder denna standard.

**Kort om PHP:s uppbyggnad**
PHP är uppbyggt av en rad olika komponenter eller moduler som tillsammans bildar en enhet.




Webbservern hanterar en inkommande förfrågan om en sida vilken förs vidare till PHP via SAPI. SAPI står för *Server Abstraction API* och utgör gränssnittet mellan PHP och webbservern i sig. Zend[7] är den fristående kärna(språkmotor) som PHP bygger på som är kodad i C. Det är här som syntaxen i PHP hanteras och tolkas till något som kan utföras på en dator. Via Zends API för externa moduler så görs anrop till dessa. De externa modulerna kan bland annat vara gränssnitt mot databaser, HTTP eller GUI-verktyg som GTK+[8].

[6] http://en.wikipedia.org/wiki/Open_Database_Connectivity
[7] http://www.zend.com/en/community/php
[8] http://gtk.php.net/
http://www.gtk.org/

## PHP - språket

PHP används oftast tillsammans med HTML och det är så vi kommer att använda det i den här kursen. Du inkluderar alltså din PHP-kod i ett HTML-dokument på samma sätt som du tidigare gjort med JavaScript. Det finns tre olika typer av öppnings/stängnings-taggar:

```
<?php
  echo "<p>Hej svejs.</p>";
?>
eller
<?
  echo "<p>Hej svejs.</p>";
?>
eller
<script language="php">
  echo "<P>Hej svejs.</P>";
</script>
```

Som ni märker är det sista alternativet närmast JavaScript i standard. Det rekommenderas dock inte att använda <script>-taggen som är avsedd att markera information som skall tolkas i webbläsaren och inte på servern vilket sker med PHP. Attributet language är dessutom borttaget för HTML 4.01 Strict.

Den tag du bör använda är den första <?php …?>. Den andra taggen fungerar vanligtvis också bra men det blir problem om man kombinerar den med XML så därför är det lika bra att vänja sig vid den första varianten. Den korta taggen kan även stängas av i servern så man måste kontrollera detta dessutom.

Kommandot echo används för att eka eller skriva ut text till webbläsaren. Inuti echo kan vi skicka med HTML-kod i form av en sträng. Alla satser i PHP måste avslutas med semikolon (;) och kallas även för instruktionsterminator. Om du inte avslutar en sats med semikolon så kommer du att få ett fel när du skall köra din PHP-kod.

Kör exemplet nedan genom att ladda upp det med filnamnet helloworld.php i din public_html-katalog på ditt konto på skolan och använd dvwebb.mah.se och "surfa" till det som om du skulle öppna en hemsida (finns även i filen helloworld_ex.php som hör till föreläsningen).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Titel</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body>
  <?php
    echo "<p><em>Hello World!</em></p>";
  ?>
</body>
</html>
```

här startar php-scriptet

echo skriver ut det inom fnuttar

här slutar php-scriptet

Ett PHP-script kan inte köras lokalt om man inte har en egen webbserver med PHP-stöd på samma dator. Ni måste alltså ladda upp ert PHP-script till en plats som har en webbadress och PHP-stöd och surfa till det med en webbläsare för att se om det fungerar.

**Escape-tecken**
Escape-tecken används för att skilja på kod och vanlig text när man vill skriva text som annars är reserverad som kod, till exempel citationstecken. För att skriva ut: Att leta runt på Internet kallas för att "surfa". Skriver man i PHP

```
echo "<P>Att leta runt på Internet kallas för att \"surfa\"</P>"
```

\-tecknet kallas för escapetecknet, det sätts framför det man ska skriva, ska man skriva just \-tecknet så skriver man \\.

**Kommentarer**

```
<?php //Detta är kommenterad text
/*Detta är kommenterad text som kan användas till mer än en rad*/
#Detta är kommenterad text
?>
```

**Att kombinera PHP och HTML**
Det finns olika sätt att blanda PHP med HTML på. Man kan börja med startagg för PHP och eka - med funktionen echo skriva ut alla HTML-taggar tillsammans med erforderlig text. Eller skriva startagg (<?php) och sluttagg (?>) runt varje PHP-kodsnutt. Exempel på de olika sätten kan du se i filerna som hör till materielet.

Alla rader som inte innesluts av skripttaggar går som output till webbservern. Detta gäller även tomma rader. Så i exemplet nedan så går först en tom rad som output till webbservern och sedan det som ekas ut i skriptet och sedan ytterligare en tom rad.

```
<?php
  echo ”Text som ekas ut efter tom rad”;
?>
```

**Variabeltyper**
I PHP finns det 8 stycken grunddata-typer: boolean, integer, float och string, Array, object, NULL, och resource. Liksom JavaScript är PHP även det löst "typat" det vill säga datatyperna för en variabel kan ändras under dess livslängd.

- Integer: allokerar oftast 32 bitar och kan representera tal mellan -2 till +2 miljarder. Storleken på Integer är dock plattformsberoende.
- Float: Reella tal även kallade flyttal, innehåller decimal eller exponent, maxstorleken är plattformsberoende. (Även kallad ”double”)
- String: Representerar bokstäver och andra tecken men det går att lagra siffror också.
- Array: Används som i andra programmeringsspråk till att lagra flera variabler. Vad som är speciellt med PHP likt JavaScript är att det tillåter att man får ha flera olika datatyper i samma vektor.
- Object: En variabeltyp som kan innehålla vilka andra datatyper som helst, men även funktioner, som i objekt även kallas för metoder. Denna datatyp används för att skapa en kopia av en så kallad klass i objektorienterad programmering.
- Boolean: datatypen som huserar booleska värden, true=1 och false=0, båda är case-insensitive.
- Null: Används för att indikera att en variabel är tom, det vill säga att den inte blivit tilldelade något värde. Man kan använda denna i till exempel jämförelseoperatorer.
- Resource: Används av speciella variabler, som innehåller en identifierare till en extern resurs. Den externa resursen kan till exempel vara en MySQL-databas. Exempel: $db = mysql_connect("localhost", "jonas", "värde11"); I exemplet skapar

funktionen mysql_connect en databas-koppling till en MySQLserver. Variabeln db blir tilldelad en identiferare till denna server och således kan den sägas vara av typen resource. PHP håller själv reda på vilka variabler av typen resource som inte används och rensar bort de tillhörande resurserna från minnet. PHP har med andra ord en automatisk så kallad garbage-collector (minneshanterare).

Grunddata-typerna kan du se mer detaljer om på:
http://www.php.net/manual/en/language.types.php

Det finns också fördefinierade konstanter[9] såsom till exempel: __FILE__ sökväg till filen du är i. PHP_OS talar om vilket operativsystem den som läser PHP-filen har.

Det finns även en uppsättning så kallade miljövariabler, till exempel:
$adress = getenv(”REMOTE_ADDR”); // surfarens IP-adress
$agent = getenv(”HTTP_USER_AGENT”); // webbläsaren

Miljövariabler kan användas till att styra olika IP-grupper till olika datorer. Om man har en så populär webbplats så att en server inte orkar serva alla användare, sätter man en dator till att omdirigera trafiken med avseende på IP-nummer till olika speglade servrar.

**Variabler i PHP**
För att deklarera en variabel skriver man $ framför variabelnamnet och tilldelar variabeln ett värde. Exempel:

```
$user = "Nils"; //variabeln user tilldelas strängen Nils
$varde = 11; //variablen varde tilldelas heltalet (integer) 11.
```

Variabelnamn får ej börja på en siffra men kan innehålla siffror. De är också case-sensitive. Variabler kan i PHP, som i andra programmeringsspråk, vara antingen globala eller lokala. Lokala variabler "lever" bara i den funktion eller del av programmet där de är deklarerade. Globala variabler "lever" i hela skriptet. I versioner över PHP 4.3 finns det möjlighet att slå av stödet för globala variabler. Man kan alltså inte räkna med att skript som implementerar globala variabler fungerar i dessa miljöer.

Fördefinierade variabler är variabler som definierats i själva programspråket. Dessa variabler kallas också "superglobals"[10] eller ”predefined variables”.

**Strängar i PHP**
Strängar kan i PHP gå över flera rader. Det är dock skillnad på om strängar skrivs inom citationstecken (”) eller apostrofer (’).

Med citationstecken ”…” gäller:

- specialtecken (de med escapetecken innan) t.ex. \t och \n konverteras
- variabler tolkas och resultatet eller värdet inkluderas
- kan innehålla apostrof ’
- citationstecken ” måste backslashas \” (Escape-tecken)

[9] http://se.php.net/manual/en/language.constants.predefined.php
http://se.php.net/manual/en/reserved.constants.php
[10] http://se.php.net/manual/en/reserved.variables.php
http://www.php.net/manual/en/language.variables.superglobals.php

Med apostrofer ’…’ gäller:

- alla tecken ingår i strängen förutom ’
- apostrofer (’) måste backslashas \’ om tecknet ska med i strängen.

Strängar innehåller alltid text; det heter egentligen textsträngar. (Det innebär att siffror i strängar ej kan bearbetas utan endast visas.)

Om en variabel deklareras såsom

```
$nummer="22";
```

så lagras variabeln av typen sträng. Däremot om en variabel deklareras såsom

```
$nummer=22;
```

så lagras variabeln av typen Integer.

Följande exempel visar hur en variabels värde kan inkluderas i en sträng. När dubbla citationstecken används så tolkas variablerna inuti som beskrivit ovan. Använd dubbla citationstecknet så blir det rätt.

```
$text = "en egen hund";
echo 'Jag är lycklig, jag har $text!<BR>';
echo "Jag är lycklig, jag har $text! <BR>";
```

Utskrift:
Jag är lycklig, jag har $text !
Jag är lycklig, jag har en egen hund !

Det första exemplet ger $text eftersom variabelnamnet inte tolkas. Allt inom ’ och ’ tolkas som en sträng. I det andra exemplet så tolkas variabelnamnet och dess värde hämtas, det blir alltså en egen hund. Jämför med följande exempel:

```
<?php
$namn = "Erik";
echo 'Hej '.$namn.'!';
echo "Hej $namn!";
echo "Hej ".$namn."!";
?>
```

Samtliga ger Hej, Erik!
. -operatorn konkatenerar (slår samman) två eller flera strängar.

**Operatorer i PHP**
Operatorer, används för att antingen tilldela, ändra eller jämföra något. Det finns i fyra typer: tilldelnings-, aritmetiska, jämförelse- och logiska operatorer. I princip fungerar detta som i JavaScript.

Exempel tilldelningsoperatorer:

```
$test1 += 1; //samma som test1=test1+1;
$test2 -= 3; //samma som test2=test2-1;
$test3 .="Hej"; //Konkatenerar eller slår ihop strängen som finns i
                //variabeln test3 med "Hej"
```

Vanliga aritmetiska operatorer som + - / *, följer standardtilldelning Exempel aritmetiska operatorer:

```
$a=$a+3;
```

Några jämförelseoperatorer:

- ==, likhet
- != inte lika med
- >större än, < mindre än
- >= större eller lika med, <= mindre lika med

Några logiska operatorer:

- && och
- || eller
- xor exklusivt eller
- ! inte

Exklusivt eller innebär att uttrycket är falskt om båda delarna är sanna.
*true xor false*
utvärderas alltså till sant medan
*true xor true*
utvärderas till falskt till skillnad från *true* || *true* som ger sant.

Operatorer finns att läsa mer om detaljerat på:
http://www.php.net/manual/en/language.operators.php

**Arrayer**
Du kan givetvis använda arrayer i PHP också. Det enklaste sättet att skapa en array är med den inbyggda funktionen array:

```
$myArray = array(1, 2, "värde");
```

Detta ger en array med paren indexnummer och värde enligt:
myArray[0] innehåller talet 1
myArray[1] innehåller talet 2
myArray[2] innehåller strängen "värde"

Du kan alltså blanda tal och strängar i en array. Det som kallas index måste inte vara ett talvärde och det måste inte börja med 0. Du kan specificera var du vill att din array skall starta och du kan använda strängar som index, eller nycklar som det då också kan kallas. Detta kallas en associativ array.

Så för att specificera både nyckel och värde när vi skapar arrayen så skriver vi:

```
$myArray = array("firstname" => "Testa", "lastname" => "Provsson");
```

Detta ger arrayen:
myArray['firstname'] innehåller strängen "Testa"
myArray['lastname'] innehåller strängen "Provsson"

Alla värden måste inte anges direkt när arrayen skapas utan du kan bygga upp din array efterhand. Om du anger ett index och tilldelar detta ett värde så skapas detta index med innehåll i arrayen. Arrayens index måste därför inte följa en viss ordning.

```
$myArray = array();
$myArray[1] = "värde i index 1";
$myArray['nbr'] = 3;
```

Det går också bra att skapa flerdimensionella arrayer genom att ange flera nycklar/index:

```
$myArray['names'][0] = "Testa";
$myArray['names'][1] = "Kolla";
```

Det finns många fler sätt att använda arrayer på än vad det är rimligt att ta upp här. Du kan läsa om arrayer på:
http://www.php.net/manual/en/language.types.array.php

När det gäller att använda arrayer i strängar så uppstår en del problem med hur fnuttar behandlas. Detta kan du läsa om på:
http://www.php.net/manual/en/language.types.string.php

## Formulär och olika sätt att skicka data

Formuläret är vårt gränssnitt mot användaren. Ett HTML-formulär kan lämna data till ett PHP-skript via två olika metoder:

- GET
- POST

I PHP så finns en funktionalitet vid namn ”register globals”. Detta innebär att alla inkommande variabler med indata automatiskt får motsvarande globala variabler i skriptet. Fördelen med detta är att man inte behöver bry sig om med vilken metod indatan skickas. Användaren kan dock manipulera indatan, speciellt med metoden GET. Detta kombinerat med goda gissningar om namnet på andra globala variabler i skriptet utgör en säkerhetsrisk. Därför så är det inte längre default att register globals är aktiverat på PHP-servern från och med PHP 4.2. I den senaste versionen av PHP (PHP 6) så är denna funktion helt borttagen.

Med egenskapen ”register globals” avslaget måste programmeraren känna till vilken metod som används. Vi kommer att arbeta med ”register globals” avslaget på den server som används för kursen då man aldrig kan förutsätta att detta är påslaget. Ofta stänger man av denna egenskap på sin server för att det utgör en viss säkerhetsrisk om det missbrukas. Detta innebär ingen egentligt begränsning av funktionaliteten utan bara att den som skriver ett skript behöver vara lite mer noggrann.

Data som man har skickat med POST eller GET från ett formulär hamnar i en array i PHP-scriptet: $_POST[’index’] eller $_GET[’index’] där index är en sträng som motsvarar formulärvariabelns namn. Detta gäller PHP5 som vi använder på kursen. Observera dock att om PHP-versionen är tidigare än 4.1.x så måste man lägga till en del: $HTTP_POST_VARS och $HTTP_GET_VARS. Om vi hade haft register globals aktiverat så hade även en global variabel med namnet index skapats.

Formulärvariabelns namn måste anges med name-attributet i formulärets HTML-kod annars kan vi inte få tag i värdet på serversidan.

Exempel för en radioknapp

```
<input type=radio name='synlighet' value=1 size=40> Superusers<br>
```

synlighet är name och det matchas sedan mot det fältet i databasen. Index i $_POST och $_GET är alltså variabelns namn: Exempelvis $_POST[’synlighet’]

**Exempel – GET-metoden**

```
<form method="get" action=sida2.php>
<input type=text name=fornamn value='skriv ditt förnamn här'>
<input type=text name=efternamn value='skriv ditt efternamn här'>
<input type=submit>
</form>
```

Via URL:en skickas förnamnet och efternamnet eftersom vi använder GET som sändmetod. Exmepelvis:
http://iweb01.mah.se/~tskral/sm7/get_sida2.php?fornamn=Testa&efternamn=Provsson

På sida2.php finns koden:

```
<?php
echo "Namnet är: ";
echo $_GET['fornamn'];
echo " ";
echo $_GET['efternamn'];
?>
```

Resultatet blir:
Namnet är: Testa Provsson

Get-metoden är den metod som används om ingen metod anges. Nackdelarna med GET metoden är att man inte kan skicka hur mycket data som helst samt att användaren ser vad man skickar. Anledningen till att man inte kan skicka hur mycket data som helst är att vissa webbservrar inte accepterar hur långa URI:er (Uniform Resource Indicator) som helst. Protokollet http i sig utgör ingen begränsning.

**URL-encode**
För att skicka variabler via URL med andra tecken än [a-z9-0_] krävs funktionen urlencode(). En länk som skickar med variabeln $ort där $ort = 'Stora Lund':

```
<a href=”file.php?ort=<?php urlencode($ort);?>”>Nästa</a>
```

Resultat: file.php?ort=Stora+Lund

För att få tillbaka ursprungstexten så används funktionen urldecode.

**Exempel – POST**
Antag att detta står i en html-fil av något slag.

```
<form method="post" action=sida2.php>
<input type=text name=fornamn value=’Kalle’>
<input type=submit>
</form>
```

I filen som hämtas när man trycker på submit (sida2.php) finns PHP-koden:

```
<?
echo "Förnamnet är: ";
echo $_POST[’fornamn’];
?>
```

POST-metoden skickar datan via STDIN (Standard Input). Det innebär att POST inte har någon överföringsbegränsning, användaren ser heller inte informationen som skickas.

När ska vilken metod då användas?

- GET rekommenderas att använda till att ta fram data från en databas eller dylikt så länge det rör sig om kortare indata för sökningen eller instruktionen. Om det inte är känslig information som skickas så kan GET användas. Användaren kan alltid enkelt kan ändra informationen som skickas med GET.
- POST rekommenderas att använda för att lägga till data till databasen eller om man skickar information som är känslig på något sätt och som man vill minska risken att användaren manipulerar indatan. Det går fortfarande att manipulera POST-data men det är svårare än att manipulera GET-data.

## Villkor i PHP

PHP har som andra programmeringsspråk funktioner för att utföra saker baserat på olika alternativ, det vill säga villkorsbaserade beslut.

### if-satsen

Exempel

```
if ($a > $b){
  echo "<P>$a är större än $b</P>";
}
else {
  echo "<P>$a är mindre än $b</P>";
}
```

{ och } fungerar som i JavaScript.

### Villkorsoperatorn

Man kan skriva if-satsen ovan på ett annat sätt:

```
<?php
$svar = ($a > $b) ? "$a är större än $b" : "$b är större än $a";
echo $svar;
?>
```

Det får samma effekt men kan uppfattas som krångligare och mer svårläst.
Syntax: villkor ? värde om sant : värde om falskt

### Upprepade if-satser - elseif

Man kan även använda if tillsammans med elseif satsen för att kunna hantera många olika fall. Exempel:

```
<?php
$laeget = "bra"; //kunde också varit en fråga till användaren
if ($laeget=="superbra")
  echo('Läget är superbra');
elseif ($laeget=="bra")
  echo('Läget är bra');
else
  echo('Läget är kast');
?>
```

**switch-statsen**
Switch-statsen hanterar också flera fall under förutsättning att villkoret är en variabel som kan utvärderas till enskilda värden. Exempel:

```
<?php
switch ($laeget){
  case ”superbra”: echo ("Läget är superbra");
    break;
  case ”bra”: echo ("Läget är bra");
    break;
  default: echo ("Läget är kast");
    break;
}
?>
```

Default utförs om inga av villkoren ovan stämmer, motsvarande sista else i if-else-koden ovan. Egentligen är switch-satsen ingenting annat än en lång rad av if() else if() else if() else… men är betydligt tydligare om man hanterar många olika tillstånd hos variabeln.

**Operatorers prioritetsordning**
Det finns en prioritetsordning i jämförelse- och de logiska operatorerna som kan vara värd att ha i åtanke när man använder dessa:

1. ! ++ --
2. / * %
3. + -
4. < <= => >
5. == === !=
6. &&
7. ||
8. = += -= /= *= %= .=
9. and
10. xor
11. or

Parenteser kan ändra prioritetsordningen så att uttrycket inom parentes beräknas först.

## Iterationer
Loopar används som bekant till att utföra repeterande uppgifter. Det finns tre olika loopar i PHP: for, while och do while.

**for-loopen**
Syntax for-loopen:

```
for(initieringsuttryck; testuttryck; ändringsuttryck)
{
//kod som ska exekveras ett antal gånger.
}
```

Exempel på att skriva ut multiplikationen mellan 1-10 och 6:

```
<?php
for ($raknare=1; $raknare<=10; $raknare++)
{
  echo("$raknare gånger 6 är ".($raknare*6));
}
?>
```

Antag att räknar-variabeln $raknare sätts av användaren i ett inmatningsfält så vi kan inte veta vad denna är när vi skriver skriptet.

```
<?php
for ($raknare; $raknare<=10; $raknare++){
  if ($raknare == 0)
    break;
  $temp=100/$raknare;
  echo("100 delat med med $raknare är: $temp<br>");
}
?>
```

Det reserverade ordet break används här för att undvika division med 0 som genererar ett fel i PHP. I detta fallet så avbryts hela loopen och inga beräkningar sker.

Om vi däremot bara vill hoppa över just det fallet när räknaren är 0 så kan vi göra på följande sätt:

```
<?php
for ($raknare; $raknare<=10; $raknare++){
  if ($raknare == 0)
    continue;
  $temp=100/$raknare;
  echo("100 delat med med $raknare är: $temp<br>");
}
?>
```

continue används här istället för att hoppa över divisionen med 0 som annars hade resulterat i ett fel. Vi hoppas alltså till nästa iteration i loopen istället för att avbryta hela loopen. Vilken av de båda som används beror givetvis av vad det är programmet ska utföra för uppgift.

**Att loopa igenom alla element i en array**

```
for ( $i = 0; $i < count($a); $i++) {
  echo $a[$i]
}
```

Metoden count returnerar antalet element i en array.

Om vektorn har numeriskt index kan som i exemplet echo $a[$i]; användas. Alternativt sätt är foreach() som kan användas om index inte är numeriskt.

```
foreach($a as $x) {
  echo $x;
}
```

Detta innebär att loopen itererar över alla element i arrayen $a och i loopen så refereras det aktuella elementet som $x.

**While-loopen och do-while-loopen**
Varje gång en while-loops villkor utvärderas som sant går den runt ett varv – en ny iteration.
Syntax:

```
while (uttryck){
  //gör någonting som också påverkar uttryck
}
```

Exempel som skriver ut 2, 4, 6, 8, o.s.v. till 24.

```
<?php
$raknare = 1;
while ($raknare <=12) {
  echo ("$raknare gånger 2 är ".($raknare*2));
  $raknare++;
}
?>
```

Do-while-loopen används när vi absolut vet att den första iterationen alltid skall ske. Syntax:

```
do {
  //gör någonting som också påverkar uttryck
} while (uttryck);//avslutas alltid med ;
```

Exempel som skriver ut 2, 4, 6, 8, …24.

```
<?php
$raknare=1;
do {
  echo("$raknare gånger 2 är ".($raknare*2);
  $raknare++;
}
while ($raknare <=12);
?>
```

Skillnaden mellan de båda är att i do-while loopen så hinner koden exekveras en gång innan villkoret kontrolleras och det är därför vi måste vara säkra på att detta kan ske utan problem.

## Server

Den server som vissa av er tidigare använt för att surfa till filer som använder JavaScript, homeweb.mah.se, skall vi inte använda för php-filer. När ni använder PHP så körs ju skriptet på servern och i vissa fall så kan ni riskera att hänga servern om ni skapar oändliga loopar. På homeweb.mah.se såär vi inte garanterade att PHP fungerar.

När du jobbar med PHP-filer så lägger du dem som vanligt i din public_html-katalog men för att testa dem så surfar du till:

http://dvwebb.mah.se/~datorid

Denna server har de funktioner vi behöver på kursen och är avsedd att användas för kurser som använder serverbaserade tekniker så vi inte stör fler användare än nödvändigt om någons skript skulle orsaka problem. Även om denna server har olika mekanismer för att minska risken att en student skulle påverka för flera så går den inte att helt eliminera.

Det kan vara av intresse att veta hur servern man använder är konfigurerad. Genom den inbyggda funktionen phpinfo() så kan du ta reda på detta . Denna genererar en sida som visar information från servern:

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

## Länkar

Du bör läsa dessa länkar och det som det refereras till i texten ovan. Inför inlämningsuppgifterna så förutsätts du ha läst de sidor som finns på php.net och som det hänvisas till då dessa ger detaljinformation som du kan behöva.

**Om variabler med indata till skriptet och säkerhetsrisken med register globals:**
http://www.php.net/manual/en/language.variables.external.php
http://www.php.net/register_globals

Något föråldrad artikel då den gäller PHP 4 men den har ett bra exempel.
http://www.onlamp.com/pub/a/php/2003/03/20/php_security.html

**Formell definition av GET- och POST-metoden:**
http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.3

## Exempelfiler

Till detta materiel så hör en mängd exempelfiler som du förutsätts bekanta dig med.