

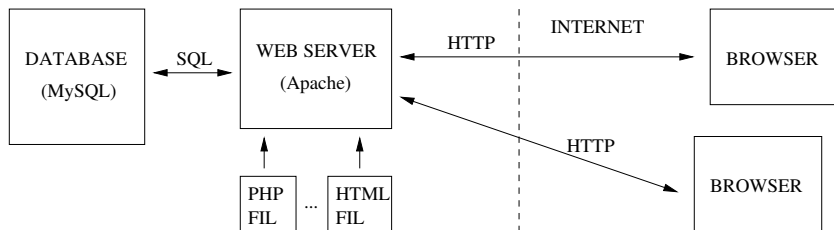
- ▶ Hvad har vi lært om webprogrammering med PHP?
- ▶ Forberedelse til eksamen
- ▶ Teknologier for sites der er programmer
- ▶ Løkker fortsat (for-løkker og ulykker)
- ▶ Funktioner
- ▶ Indbyggede funktioner
- ▶ Brugedefinerede funktioner
- ▶ Genbrug af kode med `include`-filer

Eksempelsiden virker igen:

<http://itu.dk/courses/DSDS/E2007/php>

Hvad har vi lært om webprogrammering med PHP?

Oversigt:



Sidste gang:

- ▶ Variabler, tal og strenge
- ▶ Beregninger
- ▶ if-sætninger
- ▶ while-løkker

- ▶ Lav øvelserne
- ▶ Stil spørgsmål
 - ▶ Newsgroup
 - ▶ Hjælpelærerne
 - ▶ Diverse fora, eks. <http://www.udvikleren.dk/PHP/Forum.aspx>
- ▶ **15** timer
- ▶ Find en problemstilling og løs den
- ▶ Prøv eksempler fra bogen og forelæsninger, tast selv

- ▶ Beslut om du bygger dynamiske dokumenter eller programmer med et web-interface
 - ▶ Vælg et programmeringssprog (C, PHP, Java, Standard ML, ...)
 - ▶ Vælg en programinvokeringsmekanisme (CGI-scripts, webserver-fortolkning)
 - ▶ Vælg en web-server som understøtter ovenstående tre valg
1. **Site generation:** Programmer som genererer HTML kode
 2. **Dynamic sites:** Server-fortolkede programmer som genererer HTML kode (f.eks. PHP scripts under Apache web-serveren)

Microsoft's Active Server Pages (.asp-sider):

```
<html>
...
  <%  VBScript eller JScript  %>
...
</html>
```

PHP:

```
<html>
...
  <?php PHP kode ?>
...
</html>
```

PHP (Tcl, Perl, ...)

- + *godt til at strengåndtering*
- + *hurtig udviklingscyklus (ingen oversættelse)*
- ÷ *dynamisk typet; fejl fanges først runtime*

Java

- ÷ *stort og komplekst*
- + *statisk typet; mange fejl fanges compiletime*

Standard ML (SMLserver)

- + *statisk typet; mange fejl fanges compiletime*
- + *scripts genererer med garanti velformat XHTML*
- ÷ *få Standard ML programmører*

1. CGI-scripts

- + *alle programmeringssprog kan i princippet bruges*
- ÷ *stor starttid for hvert program (skalerer dårligt)*

2. Programmer fortolket i serveren

- ÷ *kun understøttede sprog kan bruges*
- + *lav opstarttid for hvert program*

1. Microsoft's Internet Information Server (IIS)

- + *understøtter .asp sider*

2. Apache

- + *nem at udvide med moduler (til f.eks. fortolkning af PHP-kode)*

For-løkker — en kortere notation for `while`-løkker

Med `for`-løkker kan man det samme som med `while`-løkker. Hverken mere eller mindre!

Betragt programmet `love.php`:

```
$counter = 20;
while ( $counter >= 1 ) {
    echo "I love Web programming<br />";
    $counter--;
}
```

`while`-løkken kan omskrives til en `for`-løkke — `love2.php`:

```
for ( $counter = 20 ; $counter >= 1 ; $counter-- ) {
    echo "I love Web programming<br />";
}
```

Husk: `$counter--` er det samme som `$counter = $counter - 1`
`while`-løkker er ofte mere intuitive for begyndere.

En for-løkke har følgende form:

```
for ( initialisering ; betingelse ; optælling ) {  
    kommandoer ;  
}
```

Den virker sådan:

- (1) *Udfør initialisering*
- (2) *Udregn betingelse; hvis 0 (FALSK) fortsæt efter løkken*
- (3) *Udfør kommandoer (løkke kroppen)*
- (4) *Udfør optælling*
- (5) *Fortsæt med punkt 2*

Eksempler — for1.php:

Standard-løkke der udskriver ---, ---, ---, ---, ---, ---, ---, ---, ---, ---:

```
for ( $i = 10 ; $i >= 1 ; $i = $i - 1 ) {  
    echo "$i, ";  
}
```

Standard-løkke der udskriver ---, ---, ---, ---, ---:

```
for ( $i = 1 ; $i <= 10 ; $i = $i + 2 ) {  
    echo "$i, ";  
}
```

Standard-løkke der udskriver ---, ---, ---, ---, ---, ---, ---:

```
for ( $i = 1 ; $i <= 100 ; $i = $i * 2 ) {  
    echo "$i, ";  
}
```

Skriv en løkke der udskriver 64, 32, 16, 8, 4, 2, 1:

```
for ( $i = __ ; __ >= __ ; $i = __ / __ ) {  
    echo "$i, ";  
}
```

Skriv en løkke der udskriver 2, 4, 6, 8, ..., 100:

```
for ( $i = __ ; __ <= __ ; $i = $i + __ ) {  
    echo "$i, ";  
}
```

Skriv en løkke der udskriver 100, 110, 120, ..., 200:

```
for ( $i = __ ; __ <= __ ; $i = $i + __ ) {  
    echo "$i, ";  
}
```

En `for`-løkke kan altid omskrives til en `while`-løkke:

```
for ( initialisering ; betingelse ; optælling ) {  
    kommandoer ;  
}
```

omskrevet til `while`-løkke:

```
initialisering ;  
while ( betingelse ) {  
    kommandoer ;  
    optælling ;  
}
```

Lad os lave en multiplikationstabel — mul.php:

```
for ( $r = 1 ; $r <= 5 ; $r = $r + 1 ) {  
    // kør følgende for-løkke for hver række  
    for ( $s = 1 ; $s <= 5 ; $s = $s + 1 ) {  
        echo ( $r * $s );  
        echo " ";  
    }  
    echo "<br />";  
}
```

Den ydre løkke kører 5 gange, $r = 1, 2, \dots, 5$

For hvert r kører den indre løkke med $s = 1, 2, \dots, 5$

Hvor mange gange udføres multiplikationen (*) ?

Bemærk brugen af indrykning i koden — indrykning letter forståelsen!

En pænere multiplikationstabel

Lad os formatere tabellen ved brug af HTML `<table>` element — `mul2.php`:

```
<html><head><title>Multiplikationstabel</title></head>
  <body>
    <h2>Multiplikationstabel</h2>
    <table>
      <?php
        for ( $r = 1 ; $r <= 5 ; $r = $r + 1 ) {
          echo "<tr>";
          for ( $s = 1 ; $s <= 5 ; $s = $s + 1 ) {
            $tmp = $r * $s;
            echo "<td width='30' height='30'>$tmp</td>";
          }
          echo "</tr>";
        }
      ?>
    </table>
  </body>
</html>
```

Ulykker !

Hvad er der galt her:

```
for ( $i = 0 ; $i < 10 ; $i = $i - 1 ) {  
    echo "$i, ";  
}
```

Hvad er der galt her:

```
$j = 0;  
while ( $j < 10 ) {  
    echo "$j, ";  
}
```

Morale: Det er vigtigt at løkkerne terminerer — ellers afslutter scriptet ikke !

Funktioner, del 1

Funktioner åbner mulighed for at navngive en stump kode

Kode-stumpen (funktionen) kan derefter *kaldes* fra forskellige andre steder i koden

Vi har allerede set eksempler på funktionskald — `echo` er en funktion

En funktion kan tage argumenter til brug i funktionen

En funktion kan returnere en værdi til brug i koden hvorfra funktionen kaldes

Den returnerede værdi kan udskrives

```
<?php echo myfunction(); ?>
```

eller gemmes og bruges i andre sammenhæng

```
<?php $value = myfunction();  
echo $value;  
?>
```

PHP har mange indbyggede funktioner som kan kaldes fra din kode:

- ▶ Matematiske funktioner (`sin`, `cos`, ...)
- ▶ Funktioner til strenghåndtering (`str_replace`, `number_format`, ...)
- ▶ Funktioner til dato håndtering
- ▶ Funktioner til at tilgå filer på webserveren
- ▶ Funktioner til at tilgå en database, afsending af email, ...

Det er muligt selv at definere nye funktioner

Funktionen `number_format` tager fire argumenter:

1. Tallet der skal formateres
2. Antallet af decimaler i resultatet
3. En streng til at adskille de hele tal fra decimalerne
4. En streng til adskillelse af tusinder

Eksempel:

```
<?php
echo number_format(1234.5678, 2, ",", ".");
//resulterer i strengen "1.234,57"
?>
```

Funktionen `rand` tager to argumenter A og B og returnerer et tilfældigt tal mellem A og B

Eksempel:

```
<?php
$min = 1;
$max = 100;
$random = rand($min, $max);
// $random indeholder nu et tal mellem $min og $max

// Og nu udskriver vi det
echo $random;
?>
```

Brugerdefinerede funktioner

Eksempel — emph.php:

```
<?php
// Først definerer vi vores funktion
function emph ( $arg ) {
    return "<em>$arg</em>";
}

/* Herefter kan vi kalde vores funktion med
   forskellige argumenter*/
$really = emph("really");
echo "I $really like coffee!<br />";
$you = emph("you");
echo "don't $you?";
?>
```

Fordel: Vi skal kun rette et sted for at fremhæve tekst anderledes —
emph2.php:

En brugerdefineret funktion har følgende form:

```
function foo($arg_1, $arg_2, /* ..., */ $arg_n) {  
    //Det kan være vi udskriver noget  
    echo "Argument 1 = $arg_1.\n";  
  
    //Det kan være vi returnerer noget  
    return $arg_2 * $arg_n;  
}
```

Funktioner kan benytte prædefinerede argumenter

```
function foo($prefix, $name='Thomas', $postfix='ja du gør') {  
    echo $prefix.' '.$name.', '.$postfix;  
}  
foo("Du hedder");  
//udskriver Du hedder Thomas, ja du gør  
  
foo('Du hedder', 'Ole');  
//udskriver Du hedder Ole, ja du gør
```

Prædefinerede argumenter bør placeres til sidst

Hvis vi vil "ramme" \$postfix, skal vi angive \$name

```
<?php
function mypage ( $title, $body ) {
    echo "<html><head><title>$title</title></head>
        <body><h2>$title</h2> $body
        <hr><a href=\"mailto:me@mycom.com\">me@mycom.com</a>
        </body>
        </html>";
}
mypage ("About MyCom.com",
        "MyCom.com is my new company");
?>
```

Bemærk:

- ▶ Der henvises til parameteren `$title` to gange i funktionskroppen
- ▶ Der henvises til parameteren `$body` en gang i funktionskroppen
- ▶ Funktionen skriver til klienten med `echo`
- ▶ Hvordan kan vi genbruge funktionen `mypage` i andre filer?

Genbrug af kode med include-filer

Det er muligt at inkludere kode fra en fil ved brug af include-kommandoen. **Filen:** myutil.php:

```
<?php function italic ( $arg ) { return "<em>$arg</em>"; }
function mypage ( $title, $body ) {
    echo "<html><head><title>$title</title></head>
        <body><h2>$title</h2> $body
        <hr><a href=\"mailto:me@mycom.com\">me@mycom.com</a>
        </body>
        </html>";
}
?>
```

Filen mycom2.php:

```
<?php include("myutil.php");
mypage("About MyCom.com",
    "MyCom is my new company. I " .
    italic("really") . " like coffee.")
?>
```

Husk: Operatoren '.' benyttes til at sammensætte strenge

- ▶ Variable initialiseret i en funktion er *kun* tilgængelige i funktionen

```
<?php
//vi opretter en ny variabel
$b=6;
//Vi opretter en funktion
function test(){
    $a=8;
    // $b findes ikke i funktionen test(), så det her går ikke
    echo $b;
}
//Vi kalder funktionen
test();

//Vi prøver at referere en variabel der ikke findes udenfor test()
echo $a;
?>
```

Hvad udskriver ovenstående?

Scope, include

- ▶ Variable og funktioner oprettet i et eksternt dokument, kaldt via `include` er tilgængelige i det kaldende script

```
<?php
    //filen includeMe.php
    $a=100;
?>
```

```
<?php
    //filen main.php
    include("includeMe.php");
    echo $a;
    //udskriver 100
?>
```

funktioner der kalder funktioner

fun.php:

```
<?php
function f() {
    echo "L";
}
function g() {
    echo "E";
    return f();
}
function main() {
    echo "H";
    $res = g() + f();
    echo "0";
}
main();
?>
```

Ved øvelserne skal I konstruere fire små services:

- ▶ To multiplikationstabelgeneratorer
- ▶ En udvidet version af body-mass-index servicen fra sidste forelæsning
- ▶ En terningkast-service