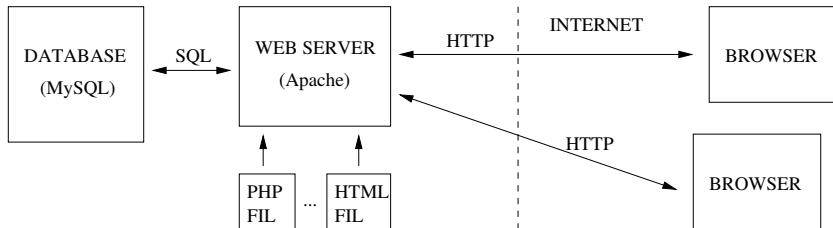


- ▶ Hvad har vi lært indtil nu?
- ▶ Kort om `include()`, igen
- ▶ Motivation for check af brugerindtastninger (formvariabler)
- ▶ Regulære udtryk (mønstre)
- ▶ PHP Funktionen `ereg`
- ▶ Eksempler på check af formvariabler
- ▶ Bibliotek af funktioner til check af formvariabler
- ▶ Anden brug af regulære udtryk
- ▶ Introduktion til øvelser

Hvad har vi lært indtil nu?

Oversigt:



En PHP-fil:

```
<html>
  <head><title>Hello World</title></head>
  <body>
    <?php echo "<b>Hello</b> ";
      echo "<i>WORLD</i>";
    ?>
  </body>
</html>
```

Indtil nu:

- ▶ Variabler, tal og strenge
- ▶ Beregninger
- ▶ `if`-sætninger og løkker
- ▶ Funktioner og kodegenbrug
- ▶ Indhentning af data fra brugere med forms
- ▶ Arrays

Kort om include()

Benytter vi en række funktioner til vores markup, skal vi kun rette ét sted. Lad os kalde den markup.php:

```
<?php
function showHeader($title){
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"...
<html lang="en">
  <head>
    <title><?php echo $title;?></title>
    <meta http-equiv="content-type" ...
  </head>
  <body>
  <?php
  }

function showFooter(){
?></body></html><?php
}
?>
```

Kort om include()

Når vi så skal bruge markup, evt til øvelserne. `includemarkup.php`

```
<?php
include("markup.php");
showHeader("Velkommen");
?>
```

Her er min besvarelse,

```
<a href='http://validator.w3.org/check?uri=referer'>
  validerer jeg?
</a>
```

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

Motivation for check af formvariabler

Uden check af brugerindtastninger er vores webprogrammer ikke robuste!

Eksempel: Veksleservice — veksle.html

```
<h2>Vekslebank</h2>
<form action="veksle.php">Indtast beløb i kroner:
  <input name="kroner"><p><input type="submit" value="Beregn Dollar
</form>
```

Filen veksle.php:

```
<h2>Vekslebank</h2>
<?php $kurs = 8.43; $gebyr = 20.0;
  $dollars = ($kroner - $gebyr) / $kurs;
  $dollars = number_format($dollars, 2, ",", ".");
  echo "For kr. $kroner modtager du \$$dollars"; ?>
<p><a href="veksle.html">Ny beregning</a></p>
```

Problem: Vi undersøger ikke om formvariablen \$kroner indeholder et tal

Hvad kan der gå galt hvis vi ikke checker formvariable?

Forskellige typer fejl kan opstå ved viderebehandling af forkerte indtastninger fra brugere:

- ▶ Kosmetiske fejl: programmet returnerer ikke-velformet html-kode til browseren
- ▶ Semantiske fejl: programmet returnerer velformet html-kode som synes rigtig, men ikke er det!
- ▶ Meget alvorlige fejl: data slettes i database fordi manglende checks kan give adgang for hackere

Ved at checke brugerindtastninger kan vi undgå de fleste typer fejl

Mange typer af formvariable

Vi har set at der er brug for at kunne afgøre om en formvariabel indeholder et tal

Der er mange andre typer af data vi kan bede brugere om at indtaste:

- ▶ Kommatal
- ▶ Email-adresser
- ▶ Mængde af tal der er begrænset af et interval (postnumre)
- ▶ URL-adresser
- ▶ Farver (red, green, blue)
- ▶ Datoer
- ▶ ...

Vi har brug for et sprog til at beskrive strenge der har en bestemt form tilfælles

Sproget vi har brug for kaldes *regulære udtryk* (eng: regular expressions). Regulære udtryk kaldes ofte for mønstre.

Syntaks for regulære udtryk, del 1

Et mønster m kan blandt andet tage følgende former:

- \cdot matches af ethvert tegn
- c matches af tegnet c ; tegnet \cdot angives som $\backslash \cdot$.
- $m_1 m_2$ sekventiel match af m_1 og m_2 . Eks: strengen 'abc' matcher mønsteret 'a.c'
- m^* matches af 0 eller flere efterfølgende forekomster af tegnsekvenser som matcher mønsteret m . Eks: strengene 'abbbbbba' og 'aa' matcher mønsteret 'ab*a'
- (m) matches af strenge som matcher m . Eks: strengen 'cababcc' matcher mønsteret 'c(ab)*cc'
- m_+ matcher et eller flere efterfølgende forekomster af tegnsekvenser som matcher mønsteret m . Eks: mønsteret 'ca+b' matches af strengen 'caaab' men ikke af strengen 'cb'

Syntaks for regulære udtryk, del 2

Et mønster m kan blandt andet tage følgende former:

$m?$ matches af 0 eller 1 forekomst af tegnsekvenser som matcher mønsteret m . Eks: strengene 'abc' og 'ab' matcher begge mønsteret 'abc?'

$m_1 \mid m_2$ matches af tegnsekvenser som matcher enten m_1 eller m_2 . Eks: mønsteret '(hej|hello)' matches af tegnsekvensen 'hej' og af tegnsekvensen 'hello'

[...] matches af tegn i klassen. Eks: mønsteret [abc1-4]* matches af tegnsekvenser bestående af tegnene a, b, c, 1, 2, 3, 4

[^...] matches af alle andre tegn end dem i klassen. Eks: mønsteret [^abc1-4]* matches af tegnsekvenser bestående af alle tegn bortset fra a, b, c, 1, 2, 3, 4. Hat-ten ^ betyder altså, alt andet end de tegn der følger

Eksempler på mønstre

- ▶ $[A-Za-zÆØÅæøå]$: matcher alle tegn i det danske alfabet
- ▶ $[0-9][0-9]$: matcher to cifrede tal der kan starte med 0
- ▶ $(hej|hello)$: matcher de to strenge `hej` og `hello`
- ▶ $((a|b)a)^*$: matcher `aa`, `ba`, `aaaa`, `baaa`, ...
- ▶ $(0|1)^+$: matcher binære tal, dvs `0`, `1`, `01`, `11`, `011101010`, ...
- ▶ $..$: matcher to vilkårlige tegn
- ▶ $(red|green|blue)$: matcher farverne `red`, `green` og `blue`
- ▶ $([1-9][0-9]^*)/([1-9][0-9]^*)$: matcher heltalsbrøker, f.eks. `1/8`, `32/5645`, `45/6`, ...

Matches strengen `012/54` af det sidste mønster?

Matches strengen `2/0` af det sidste mønster?

PHP Funktionerne `ereg` og `eregi`

Den indbyggede funktion `ereg` kan benyttes til at afgøre om et mønster m matcher en delstreng i en streng s .

Et kald `ereg(m,s)` returnerer 1 (dvs. SAND) hvis mønsteret m matcher en delstreng i s . Ellers returneres 0 (dvs. FALSK).

Hvis mønsteret m startes med '^' må ingen tegn forekomme før delstrengen i s .

Tilsvarende, hvis mønsteret m slutes med '\$' må ingen tegn forekomme efter delstrengen i s .

Eksempler:

Funktionskald		Resultat
<code>ereg('[0-9]+'</code> , "aa38AA")	→	1
<code>ereg('^ [0-9]+'</code> , "aa99")	→	0
<code>ereg('^ [0-9]+'</code> , "77AA")	→	1
<code>ereg('^ [0-9]+\$'</code> , "aa87AA")	→	0

Funktionen `eregi` fungerer ligesom `ereg` men er "case-insensitive"

Bemærk:

- ▶ Til check af brugerindtastninger vil vi oftest benytte os af tegnene \wedge og $\$$ først og sidst i mønsteret
- ▶ Til opskrivningen af mønstre benytter vi os af muligheden for at indkapsle strenge med '...' for at undgå den specielle betydning af tegnet $\$$ i strenge på formen "..."

Eksempler:

Funktionskald

		Resultat
<code>ereg('[a-zA-Z]+' , "Allan Hansen")</code>	→	--
<code>ereg('^ [a-zA-Z]+\$' , "Ulla Jensen")</code>	→	--
<code>ereg('^ [a-zA-Z]+\$' , "")</code>	→	--
<code>ereg('^ [0-9] [0-9]-[0-9] [0-9]-[0-9]+\$' , "12-22-1969")</code>	→	--
<code>ereg('^ [0-1] [0-9]-[0-3] [0-9]-[0-9]+\$' , "31-12-02")</code>	→	--
<code>ereg('^ (red green blue)\$' , "red")</code>	→	--
<code>ereg('^ (red green blue)\$' , "redblue")</code>	→	--
<code>ereg('^ (a bb)*b\$' , "bbab")</code>	→	--
<code>ereg('^ (a bb)+b*\$' , "b")</code>	→	--

Eksempel: webprogrammet monster.php:

For at eksperimentere med mønstre kan vi bygge en mønster matcher — monster.php:

```
<h2>Mønster matcher</h2>
<?php
$r=$_REQUEST['r'];
$s=$_REQUEST['s'];
if ( $r != "" ) {
    if ( ereg("^$r\\$", $s) ) {
        echo "Success: Mønsteret $r matcher strengen $s<br />";
    } else {
        echo "Fejl: Mønsteret $r matcher IKKE strengen $s<br />";
    }
}
echo "<form action='monster.php' method='post'>
<p>Mønster: <input type='text' name='r' value='$r'></p>
<p>Streng: <input type='text' name='s' value='$s'></p>
<p><input type='submit' value='Check'></p>
</form>";
?>
```

Forbedring af Vekslebank ved check af formvariable veksle2.php

Vi kan forbedre vores veksleservice ved at checke brugerindtastning

```
<?php
$kroner=$_REQUEST['kroner'];
if ( ereg('^0|([1-9][0-9]*)$', $kroner) ) {
    $kurs = 8.43; $gebyr = 20.0;
    if ( $kroner > $gebyr ) {
        $dollars = ($kroner - $gebyr) / $kurs;
        $dollars = number_format($dollars, 2, ",", ".");
        echo "For kr. $kroner modtager du \$$dollars";
    } else {
        echo "Du kan ikke veksle et beløb mindre end gebyret!";
    }
} else {
    echo "Gå tilbage og indtast et tal!";
} ?>
```

Spørgsmål: Hvordan reagerer programmet på forskellige former for indtastninger?

Funktion til check af Email-adresser

For at checke om en indtastet email-adresse har den forventede form kan vi benytte mønsteret:

```
[a-zA-Z][0-9a-zA-Z._]*@[0-9a-zA-Z._]+
```

Mønster til at tjekke email-adresser:

```
function chk_email ( $email ) {  
  if( ereg('^[a-zA-Z][0-9a-zA-Z._]*@[0-9a-zA-Z._]+$', $email)== 0){  
    error("Du skal indtaste en email-adresse");  
  }  
}
```

Bemærk:

- ▶ Funktionen garanterer ikke at email-adressen eksisterer!
- ▶ Vi benytter en generel funktion `error` til at udskrive en fejlmeddelelse:

```
function error ( $msg ) {  
  echo "<html><body><h2>Fejl: $msg</h2></body></html>";  
  exit; // Undgå at fortsætte scriptet!  
}
```

Bibliotek af funktioner til check af formvariable

Vi kan konstruere en fil `formvars.php` indeholdende en række funktioner til check af formvariable

Herefter kan vi inkludere filen (med `include()` funktionen) i de filer der benytter formvariable

Derved bliver det meget nemt at checke formvariable!

Eksempel: — `veksle3.html`

```
<?php include ("formvars.php");
$kroner=$_REQUEST['kroner'];
chk_heltal($kroner);
    $kurs = 8.43; $gebyr = 20.0;
    if ( $kroner > $gebyr ) {
        $dollars = ($kroner - $gebyr) / $kurs;
        $dollars = number_format($dollars, 2, ",", ".");
        echo "For kr. $kroner modtager du \$$dollars";
    } else {
        echo "Du kan ikke veksle et beløb mindre end gebyret!";
    }
?>
```

Bibliotek af funktioner til check af formvariable — fortsat

Lidt af filen `formvars.php`:

```
function error ( $msg ) {
    echo "<html><body><H2>Fejl: $msg</h2></body></html>";
    exit; // Undgå at fortsætte scriptet!
}

function chk_email ( $email ) {
    if(ereg('^[a-zA-Z][0-9a-zA-Z\._]*@[0-9a-zA-Z\._]+$', $email)== 0){
        error("Du skal indtaste en email-adresse");
    }
}

function chk_heltal ( $tal ) {
    if ( ereg('^0|[1-9][0-9]*$', $tal) == 0 ) {
        error("Du skal indtaste et tal");
    }
}
```

Regulære udtryk kan benyttes til andet end verificering af brugerindtastninger

Eksempler:

- ▶ Søgning efter mønster i en tekst
- ▶ Søg-og-erstat i en tekst

Indhentning af data fra fremmede websites: Indenfor webteknologi kan regulære udtryk benyttes til at finde relevante oplysninger i tekster der hentes automatisk på nettet, f.eks:

- ▶ Dagens nyheder fra Reuters
- ▶ Dollarkursen
- ▶ Aktiekurser
- ▶ Vejret
- ▶ Pollental

Funktionen `file_get_contents($url)`; kan benyttes til at hente en fremmed webside ind i en streng i PHP:

```
<?php
$html = file_get_contents("http://google.com/");
?>
```

Sammen med et regulært udtryk (et mønster) kan den indbyggede PHP-funktion `ereg` bruges til at klippe indformation ud af en webside indeholdt i en streng.

Funktionen `ereg($p,$txt,$result)` kan tage tre argumenter:

- ▶ `$p` er et mønster til afgrænsning af hvilken tekst der skal klippes ud
- ▶ `$txt` er teksten mønsteret sammenlignes mod
- ▶ `$result` er et resultat-array indeholdende udklippet tekst efter kaldet til `ereg`

Eksemplet `popclock.php` henter data fra `www.census.gov`

```
<html><head><title>World Population Watch</title></head>
<body>
<h1>World Population Watch</h1>
<?php
    $content = file_get_contents(
        "http://www.census.gov/cgi-bin/ipc/popclockw");
    eregi("<div id=\"worldnumber\">([0-9,]+)</div>",
        $content,$result);
    echo "Der er $result[1] mennesker i verden.";
?>
</body>
</html>
```

Ved øvelserne skal I:

- ▶ Øve regulære udtryk
- ▶ Konstruere en forbedret version af body-mass-index web-servicen
- ▶ Konstruere en dollarkurs web-service der henter dollarkursen fra et fremmed web-site og tilbyder omregning af kroner til dollars og dollars til kroner.
- ▶ Selv finde på anvendelig automatisk udtrækbar information på nettet og implementere en web-service der benytter sig af denne information — vejret, børskurser, pollental, ...