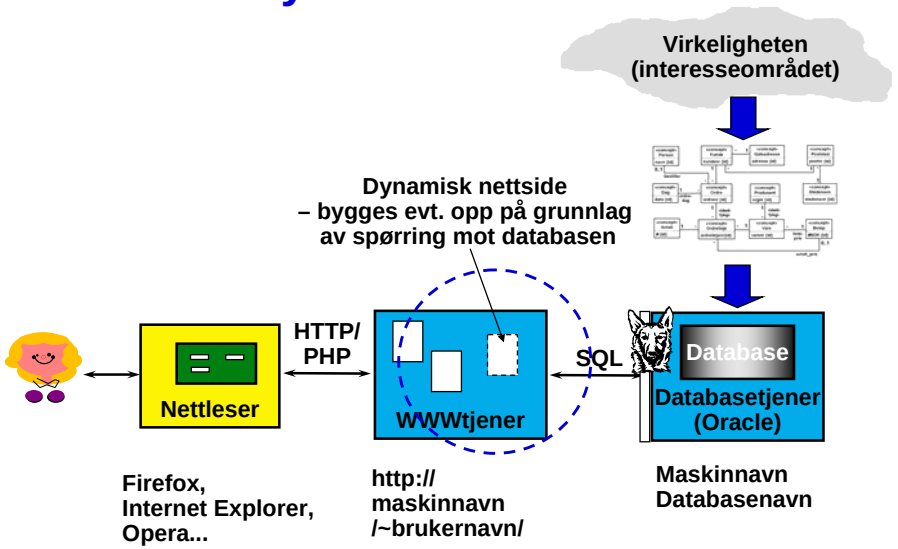
Webformularer, PHP og databaser – et crash-kurs

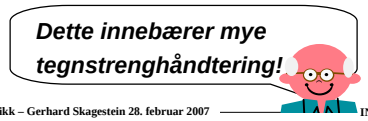
Jfr. “Fra kjernen og ut, fra skallet og inn” Appendix B

Systemarkitektur



Hva skal applikasjonsprogrammet gjøre?

- Ta i mot data fra brukergrensesnittskjermbildet
- Finne ut hva som skal gjøres – og gjøre det
- Hvis det som skal gjøres omfatter en spørring mot eller en oppdatering av en database:
  - Bygg en egnet SQL-kommando
  - Åpne databasen, send SQL-kommandoen til databasen, ta i mot resultatet, lukk basen
- Utføre beregninger etc. (hvis nødvendig)
- Bygg opp nettsiden som skal vises frem i brukergrensesnittet, i form av en XHTML-tekst



PHP

- PHP er et skriptspråk designet for å kunne utvikle dynamiske nettsider.
- PHP kan i en nettside brukes side om side med XHTML.
- PHP er ”open source”.
- PHP: PHP Hypertext Preprocessor (et rekursivt akronym ☺)
- PHP-nettsted: http://www.php.net

Vanlig XHTML-fil

På filen ~inf1050/php/helloworld.html

```
<html>
<head>
<title>PHP Hello World</title>
</head>
<body>
<h1>Hello World</h1>
<p>Hello World</p>
</body>
</html>
```

For enkelhets skyld utelater vi de innledende XHTML-besvergelsene inntil videre!

XHTML-fil med innbakt PHP

På filen ~inf1050/php/helloworld.php

```
<html>
<head>
<title>PHP Hello World</title>
</head>
<body>
<h1>Hello World fra PHP</h1>
 <?php print("<p>Hello World</p>"); ?>
</body>
</html>
```

Hva skjer?

Når vi gjennom en nettleser aksesserer en nettside med PHP-kode, vil følgende skje:

- **Webjeneren mottar forespørsel fra klienten**
- **Nettsiden letes opp på webtjeneren**
- **Webtjeneren utfører instruksjonene i PHP-koden, og eventuelle "utskrifter" blir integrert i nettsiden (dette forutsetter at webtjeneren "skjønner" PHP)**
- **Nettsiden sendes tilbake over nettet til nettleseren**
- **For nettleseren vil siden se ut som en helt vanlig XHTML-kodet side**

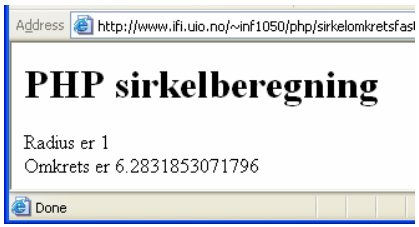
XHTML-fil med innbakt PHP

På filen ~inf1050/php/sirkelomkretsfast.php

```
<head>
    <title>PHP sirkelberegning</title>
</head>
<body>
    <h1>PHP sirkelberegning</h1>
    <p>
    <?php
     define("PI",3.1415926535897932);
     $radius = 1.0 ;
     print("Radius er ".$radius);
    ?>
    <br />
    <?php
     $omkrets = 2 * $radius * PI;
     print("Omkrets er ".$omkrets);
    ?>
    </p>
</body>
</html>
```

Generert XHTML-fil

```
<html>
<head>
    <title>PHP sirkelberegning</title>
</head>
<body>
    <h1>PHP sirkelberegning</h1>
    <p>
    Radius er 1    <br/>
    Omkrets er 6.2831853071796</p>
</body>
</html>
```



PHP-språket

- **Definere konstanter:**
  define("PI",3.1415926535897932);
  Det er vanlig å bruke store bokstaver for konstant-navn.
- **Variable:**
  Variable er *dynamisk typet*
  – en variabel kan tilordnes en verdi av en hvilken som helst type
  Variabelnavn begynner alltid med tegnet $.
  Variablen opprettes første gang den nevnes.
  Tilordning skjer ved hjelp av tilordningsoperatoren =
  Eksempel: $radius = 1.0;
- **Aritmetiske uttrykk:**
  Eksempel: 2 * $radius * PI;
- **Skjøte sammen tekststrenger:**
  Sammenskjøtingsoperatoren . (punkt)

PHP-språket

```
<?php
    define("PI",3.1415);              // definisjon av en konstant
    $radius = 1.0;                    // deklarer og initialiser $radius
    print("Radius er ".$radius);      // ”skriv” en streng
?>

<?php
    $omkrets = 2 * $radius * PI;      // beregn $omkrets
    print("Omkrets er ".$omkrets);    // ”skriv” en streng
?>
```


PHP-språket – IF-setninger

IF-setninger

```
if ($antallKronblader % 2 == 1){
  print ("Elsker!");
}
```

```
  if ($antallKronblader % 2 == 1){
  print ("Elsker!");
} else {
  print ("Elsker ikke...");
}
```

== er en sammenlikningsoperator (comparison operator).
Se en komplett liste på
http://no2.php.net/manual/en/language.operators.comparison.php

PHP-språket – Løkker

FOR-løkker

```
for ( $i = 1; $i <= $antall; $i++ ) {
  .
  .
}
```

WHILE-løkker

```
while ($i <= $antall) {
  .
  .
  $i = $i + 1;
}
```

PHP bruker assosiative arrayer

- Elementene i et assosiativt array består av nøkkel-verdi-par (”key-value-pairs”)
- Som nøkler kan brukes heltall (analogt med indekser i andre språk) og tekststrenger.
- Array med heltallsnøkler (indekser), eksempel:

```
$mittarray = array(0 => 6, 1 => 13, 2 => 'Per', 3 => 'Gro', 4 => 3.14);
$mittarray[5] = 2.71;
$element3 = $mittarray[3];   // tilordner verdien ’Gro’til $element3
```

Dersom det ikke er gitt noen eksplisitt nøkkel, brukes eksisterende maksimumsverdi + 1. Initialverdien er 0. Vi kan derfor også skrive

```
$mittarray = array(6, 13,'Per','Gro', 3.14);
```

- Array med tekststrenger som nøkler, eksempel:

```
$mittarray = array('etternavn'=>'Dal', 'fornavn'=>'Gro');
$mittarray['adresse'] = 'Nygaten 123' ;
$fornavn = $mittarray['fornavn']; // tilordner verdien 'Gro'
```

- Array med både heltall og tekststrenger som nøkler, eksempel:

```
$mittarray = array(6, 13, 'fornavn'=>'Per', 3.14);
```

var_dump($mittarray) gir:
array(4) { [0]=> int(6) [1]=> int(13) ["fornavn"]=> string(3) "Per" [2]=> float(3.14) }

Tekststrenger

- Tekststrenger avgrenses med apostrof ' eller anførselstegn "
  - Ved bruk av ' bearbeides ikke tekststrengen
  - Ved bruk av " bearbeides tekststrengen, for eksempel konverteres \n til symbolet for linjeskift
- Hvis tekststrengen inneholder avgrensningssymbolet, må ”escape-symbolet” \ settes inn i forkant:

```
print("\" Jeg elsker deg!\", sa han");
```

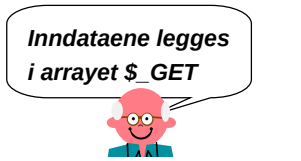
- For å unngå ”escape-symbolet”, bruk (hvis det lar seg gjøre) avgrensningssymboler som ikke forekommer i tekststrengen:

```
print("'Jeg elsker deg!', sa han");
print('"Jeg elsker deg!", sa hun');
```


Hente parametre fra brukeren

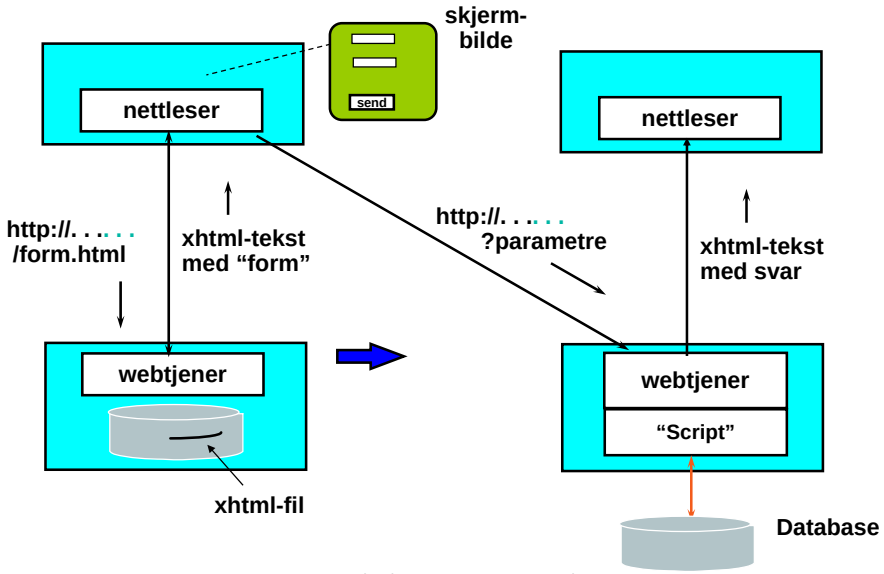
På filen ~inf1050/php/sirkelomkrets.php

```
<html>
<head>
    <title>PHP sirkelberegning</title>
</head>
<body>
    <h1>PHP sirkelberegning</h1>
    <p><?php
        define("PI", 3.1415926535897932);
        $radius = $_GET['radius'];
        print("Radius er ".$radius);
    ?>
    <br />
    <?php
        $omkrets = 2 * $radius * PI;
        print("Omkrets er ".$omkrets);
    ?></p>
</body>
</html>
```

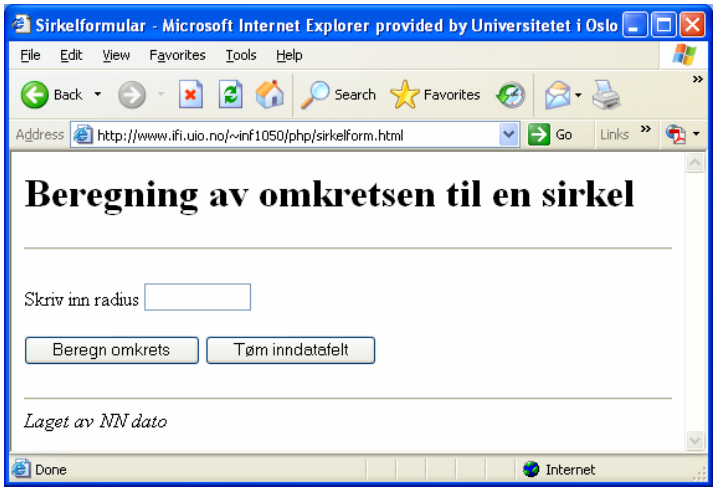


OBS!

Webformularer, parametre og scripts



Et enkelt webformular



XHTML-kode for webformularet – GET

På filen ~inf1050/php/sirkelform.html

```
<head>
<title>Sirkelformular</title>
</head>

<body>
<h1>Beregning av omkretsen til en sirkel</h1>
<hr />
<form method="get" action="sirkelomkrets.php">
    <p>Skriv inn radius <input type="text" size = "10" name="radius"/></p>
    <p><button type="submit" name="submit">Beregn omkrets</button>
    <button type="reset" name="reset">Tøm inndatafelt</button></p>
</form>
<hr />
<address>Laget av NN dato</address>
</body>

</html>
```

sirkelomkrets.php

```
$radius=$_GET['radius'];
print("Radius er ".$radius);
```

…sirkelomkrets.php?radius=…

XHTML-kode for webformularet – POST

På filen ~inf1050/php/sirkelformpost.html

```
<head>
<title>Sirkelformular</title>
</head>
<body>
<h1>Beregning av omkretsen til en sirkel</h1>
<hr/>
<form method="post" action="sirkelomkretspost.php">

<p>Skriv inn radius<input type="text" size = "10" name="radius"/></p>

    <p><button type="submit" name="submit">Beregn omkrets</button>
    <button type="reset" name="reset">Tøm inndatafelt</button> </form>
<hr />
<address>Laget av NN dato</address>
</body>
</html>
```

sirkelomkretspost.php

```
$radius=$_POST['radius'];
print("Radius er ".$radius);
```

Inndataene legges i arrayet $_POST

…sirkelomkretspost.php

XHTML-formular og PHP på samme fil

På filen sirkelberegning.php:

```
<?php
    define("PI",3.1415926535897932);
    if(isset($_POST['radius'])) $radius = $_POST['radius']; else $radius = 0;
    $omkrets = 2 * $radius * PI;
    $resultat = "Omkrets er ".$omkrets;
?>
<html>
<head>
<title>PHP sirkelberegning</title>
</head>
<body>
<h1>Beregning av omkretsen til en sirkel</h1>
<hr />
<form method="post" action="sirkelberegning.php">
<p>Skriv inn radius <input type="text" size = "10" name="radius" value="<?php print($radius);?>" />
    <br />
    <button type="submit" name="submit">Beregn omkrets</button>
    <button type="reset" name="reset">Tøm inndatafelt</button> </p>
</form>
<p> <?php print($resultat); ?> </p>
<hr />
<address>Laget av NN dato</address>
</body>
</html>
```

Husk at php-koden blir kjørt før brukeren har fått mulighet til å se skjermbildet! Derfor isset-testen!

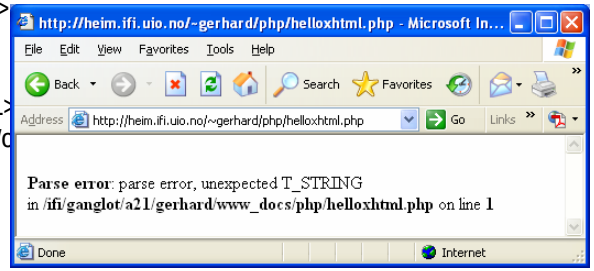
XHTML-formular og PHP på samme fil - resultat av kjøring

PHP og XHTML

Vær oppmerksom på at PHP-prosessorer kan mistolke de innledende XML besvergelser i XHTML-filer! Her et eksempel:

På filen helloxhtml.php:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3c.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<title>PHP Hello World</title>
</head>
<body>
<h1>Hello World fra PHP</h1>
    <?php print("<p>Hello Wo
</body>
</html>
```



PHP og XHTML (forts.)

Skriv derfor ut xhtml-besvergelsene med php!

På filen helloxhtml1.php:

```
<?php
print('
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3c.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns = "http://www.w3.org/1999/xhtml">
');
?>
<head>
<title>PHP Hello World</title>
</head>
<body>
<h1>Hello World fra PHP</h1>
    <?php print("<p>Hello World</p>"); ?>
</body>
</html>
```

Sessions

Hvordan overføre data fra en klient-tjener-interaksjon til den neste:

- **Åpne en "session" med funksjonskallet** session_start( );
- **Meld inn variable hvis verdier skal overføres:**
  session_register('bruker');
  session_register('passord');
- **Gjør variablene og deres verdier tilgjengelig i andre tjenerprogrammer med**
  session_start( );
- **En "session" varer til nettleseren tas ned – eller til "timeout"**
- **Dataene ligger på et beskyttet område på tjeneren**
  – pekeren til dette området overføres ved hjelp av en "cookie"

PHP mot databaser

- De fleste skriptspråk tilbyr god støtte for kobling mot databaser
- PHP har støtte for bl.a. MySQL og Oracle
- Innebygde funksjoner for kobling mot Oracle
  - OCILogon
  - OCILogoff
  - OCIExecute
  -

Kjekt å ha – på db.inc

- `authenticate()`
  setter opp en dialogboks der brukeren blir invitert til å skrive inn brukernavn og passord. Verdiene legges i $bruker og $passord som er registrert i en "session"
- `$conn = baseLogon($bruker, $passord)`
  logger inn i databasen IFIORA som brukeren $bruker med $passord. Funksjonen vil returnere en kobling mot databasen ($conn).
- `baseLogoff($conn)`
  vil stenge koblingen $conn og logge ut av databasen.
- `$stmt = baseQuery($conn, $query)`
  sender SQL-spørringen $query til databasen med koblingen $conn. Returnerer en peker til en buffer med resultatet av spørringen ($stmt).
- `byggUpdateQuery, byggInsertQuery, byggDeleteQuery`
  funksjoner for å bygge SQL-spørringer (se dokumentasjon)

Kjekt å ha – på gui.inc

- `$html = function lagSelectMeny($nokkel, $nrows, $results)`
  returnerer XHTML-kode for en nedtrekksmeny
- `$html = function visTabell($stmt, $pkattributter)`
  returnerer XHTML-kode for å vise fram tabellen som ligger i spørreresultatet `$stmt`.
- `$html = function visTabellMedLink($stmt, $pkattributter, $hreffil)`
  returnerer XHTML-kode tilsvarende `visTabell`, men med en ekstra kolonne med en aktiv link til en web-side `$hreffil`.
- `$html = function lagOppdateringsformular($stmt, $pkattributter)`
  returnerer XHTML-kode som lager et formular som gjør det mulig å endre, legge til, eller fjerne linjer i tabellen som ligger i spørreresultatet `$stmt`.
- `$html = function lagInnleggingsformular($stmt, $pkattributter)`
  returnerer XHTML-kode som lagOppdateringsformular, men med tomme felter

Eksempel på SQL-spørring med etterfølgende fremvisning

```
<?php
include "inc/db.inc";
include "inc/gui.inc";
session_start( );

$query = "SELECT fylkenr, fylkenavn FROM Fylke ORDER BY fylkenavn";
// print($query);
$conn = baseLogon($bruker, $passord);
// Sender query til databasen, resultatet legges i $stmt//
$stmt = baseQuery($conn, $query);
baseLogoff($conn);

// formater en vakker html-tabell av resultatet  //
$html = visTabell($stmt, array('fylkenr'));
print($html);
?>
```

Om feil og sikkerhet

- Eksempelprogrammene forutsetter ”happy day scenario”
- Fullstendig kontroll av alle mulige feilsituasjoner
  med tilhørende diagnosemeldinger
  vil kreve atskillig mer programkode
- Den viktigste feilkilden (og den største sikkerhetsrisikoen) ligger imidlertid i gale data fra brukergrensesnittet.
- Derfor: Sjekk disse!
  - Ser tabellnavn ut som tabellnavn ?
  - Ser numeriske verdier ut som numeriske verdier ?
  - Finnes det umotiverte – – (dvs. Oracle SQLkommentartegn) ?

  se www.php.net/manual/en/security.database.php

PHP og objekter

- PHP 5 har klasser og objekter tilsvarende Java. Eksempel:

```
<html>
<head>
    <title>Classtest</title>
</head>
<body>
<?php
// Declare a simple class
  class TestClass {
            const PI = 3.1415926 ;
            private $foo;
            public function __construct($foo){
             $this->foo = $foo; // -> tilsvarer Java . (sin)
            }
            public function __toString() {
            return $this->foo;
            }
  }
$class = new TestClass('Hello');
print ($class);
print (TestClass::PI); // Scope resolution operator (Paamayim Nekudotayim)
?>
</body>
</html>
```

Hebraisk for dobbeltkolon

Se dokumentasjon på
http://www.php.net/manual/en/language.oop5.php

Et godt råd til slutt

- Noen fakta:
  - PHP-programmer utføres uten forutgående kompilering.
  - Variabler deklareres ikke,
    de opprettes første gang de nevnes.
  - Variabler er svakt typet.
- Konklusjon:
  - Det meste er tillatt, men ikke nødvendigvis riktig!
- Den vanligste reaksjonen fra et feilaktig PHP-program er at overhodet intet skjer!
- Moral:
  - Endre og teste i meget små skritt
    – da vet du hvor feilen er!