PHP for eArkivarer

Trondheim 25. og 26. mars

KDRS
Kommunearkivinstitusjonenes
Digitale RessursSenter

PHP for eArkivarer

Velkommen og oversikt

KDRS
Kommunearkivinstitusjonenes
Digitale RessursSenter

Innhold

- To dager med PHP-skripting
- Ingen kan forvente at du blir en ekspert etter dette kurset
  - Innblikk i hvordan det er å jobbe med skripting
  - Vise de grunnlegende teknikkene som trengs
- Unngå at du faller av
  - Alle eksempler kan lastes ned fra:
  -
- Hvis du faller av så kan jeg hjelpe deg i pausen
- Viktigere å følge med hva som skjer enn å skrive PHP

Timeplan Dag 1

Tid	Team	Beskrivelse
10.00 – 10.10	Velkommen	Velkommen og gjennomgang av kursinnhold
10.10 – 10.55	PHP og skripting	Introduksjon til skripting språket PHP, oppbygging og kjøring av et PHP skript, hva PHP kan brukes til og hvordan bruke det.
11.10 – 12.00		
12.00 – 12.45	Lunsj	
12.45 – 13.45	Oppkobling til en database	Hvordan koble til en database med PHP og håndtering av feil
14.00 – 14.50	PHP og SQL	Hvordan utføre en spørring mot en database med PHP og håndtering av data
15.05 – 16.00	Fra SQL til XML	Hvordan konvertere data fra en spørring til XML og 'rensing' av innhold
16.00 – 17.00		

Timeplan Dag 2

Tid	Team	Beskrivelse
09.00 – 09.50	PHP og XML	Riktig måte å håndtere XML-dokumenter i PHP (skriving og lesing) og forskjellige tilnærminger (DOM og SAX) og API'er for håndtering av XML-dokumenter
10.10 – 10.55		
11.10 – 12.00		
12.00 – 12.45	Lunsj	
12.45 – 13.45	Fra XML til SQL	Hvordan tar vi innholdet i en XML-fil og laster det opp til en tabell
14.00 – 14.50		
15.05 – 15.50	Uttrekk i ADDML	Hvordan lagre en tabell i ADDML format

PHP for eArkivarer

PHP og skripting

Dag 1 Time 1 og 2

KDRS
Kommunearkivinstitusjonenes Digitale RessursSenter

Innhold

- Introduksjon til skriptingspråket PHP
- Hva PHP brukes til
- Oppbygging av et skript
- Kjøring av et PHP skript
- Hvordan bruke PHP

Hva er PHP

- PHP står for *Hypertext Preprocessor*
- Brukes ofte til å generere innhold til websider
- Lettere å lære seg en skriptingspråk enn en programeringspråk
- Fri programvare, gratis og mange utviklingsverktøy
- En PHP fil har filtype *.php*

Hva PHP brukes (eArkiv) til

- PHP kan brukes til
  - *Jobbe med filer*
    - opprette eller åpne for så å skrive til eller lese fra
  - *Jobbe med databaser*
    - Legge til data
    - Slette data
    - Oppdatere data
  - Håndtere skjemadata via nettleser
  - Kontrollere tilgang til data via nettleser
  - Lage en webside (HTML-fil)

En PHP-fil

```
<?php

 // PHP-koden her

?>
```

PHP-kommentarer

- Kommentarer er viktig metadata om skriptet
  - Hvem som har skrevet det, hvorfor, hvordan
  - Nyttig for å forklare andre som leser skriptet ditt senere om tenkeprossesen din
- Også en nyttig utviklingsverktøy for å slå deler av skriptet ditt ***av*** og ***på*** for å teste underveis

PHP-kommentarer

```php
<?php

  // Hele linjen er en kommentar

  /*
     Her er kommentaren over
     flere linjer
  */

?>
```

PHP-Intro

- En PHP-fil består av en eller flere kommandoer
  - Nesten litt sånn hvordan MySQL gjør noe når du gir en kommando som «SELECT * FROM TABELL»
- PHP forstår mange kommandoer
  - Alle kommandoer avsluttes med en **;**
- Når vi gir PHP-programmet et skript med mange kommandoer så vil PHP-programmet utføre de kommandoene

```
<?php
   // Min første PHP-fil
   print "Hei og velkommen til PHP-kurs";
?>
```

*Vi navngir denne filen som **hei.php***

Variabler

- Variabler brukes for navngi en beholder for informasjon
- Vi kan lage variabler for feks
  - en IP-adresse, en databasenvavn, brukernavn
- Variabler begynner med dollartegnet
  - $IPAdresse, $databasenavn, $brukernavn
- Tildele verdier til variabler
  - $IPAdresse = "127.0.0.1";
  - $databasenavn = "noark5";
  - $brukernavn = "noarkBruker";

Variabler

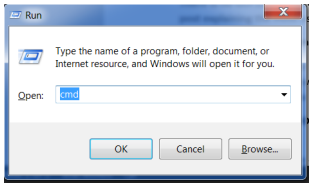
```php
<?php

    $IPAdresse = "127.0.0.1";
    $databasenavn = "noark5";
    $brukernavn = "noarkBruker";

    print "Vi skal koble til maskinen (" . $IPAdresse . ")";
    print "Vi skal koble til databasen (" . $databasenavn . ")";
    print "Vi skal koble med bruker (" . $brukernavn . ")";
?>
```

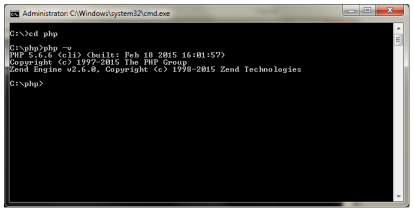
*Vi navngir denne filen som **variabler.php***

Oppgave

- Vi kjører en PHP-fil fra kommando linjen
- Vi må endre gjeldende mappe til mappen PHP-filene er i

1. 

2. 

3. 

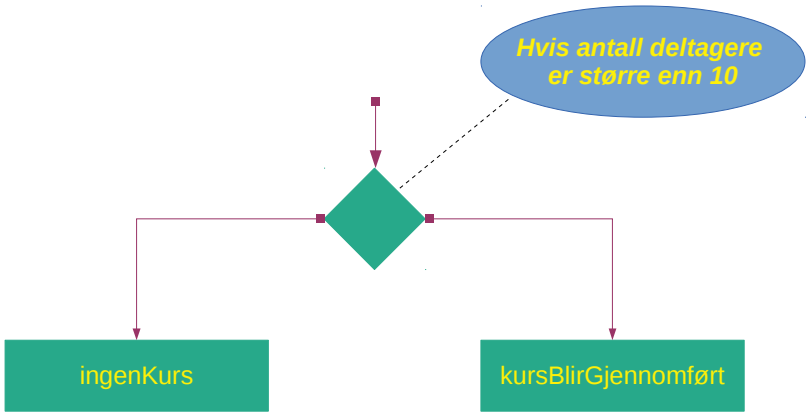
Variabler

```php
<?php

    $IPAdresse = "127.0.0.1";
    $databasenavn = "noark5";
    $brukernavn = "noarkBruker";

    print "Vi skal koble til maskinen (" . $IPAdresse . ")" . PHP_EOL;
    print "Vi skal koble til databasen (" . $databasenavn . ")". PHP_EOL;
    print "Vi skal koble med bruker (" . $brukernavn . ")". PHP_EOL;
?>
```


Datatyper

- PHP støtter flere datatyper men du trenger ikke å angi datatypen til en variabel
  - Streng (string)
    - *$kursNavn = "PHP Kurs" ;*
  - Heltall (integer)
    - *$antallDeltagere = 14;*
  - Desimaltall (float)
    - *$kursKostnad = 2400.25;*
  - Boolsk (boolean)
    - *$kveldsKurs = false;*

Oppbygging av logikk



If

```php
<?php

   $antallDeltagere = 11;

   if ($antallDeltagere >= 10) {
      print "Kurset gjennomføres";
   }
?>
```

*Vi navngir denne filen som **if.php***

If ... else

```php
<?php

   $antallDeltagere = 9;

   if ($antallDeltagere >= 10) {
      print "Kurset gjennomføres";
   }
   else {
      print "Ikke nok påmeldte";
   }
?>
```


while

- En while-løkke kan brukes til å utføre en eller flere kommandoer et gitt antall ganger
  - Kommandoer utføres så lenge testen i *while ()* ikke ansees å være sant (true)
- Typisk bruk er i en kobling til en database
  - Så lenge det er flere rader av data, hent data fra databasen

```php
<?php

   $antallDeltagere = 0;

   while ($antallDeltagere <= 10) {
      print "Antall deltagere påmeldt " . $antallDeltagere . PHP_EOL;
      $antallDeltagere = $antallDeltagere + 1;
   }

?>
```

*Vi navngir denne filen som **while.php***

for

- En for-løkke kan også brukes på samme måte en while-løkke
  - Brukes typisk når du ønsker å utføre noen kommandoer et bestemt antall ganger

```php
<?php

    for ($antallDeltagere= 0;
           $antallDeltagere <= 10;
              $antallDeltagere = $antallDeltagere + 1) {

    print "Antall deltagere påmeldt " . $antallDeltagere . PHP_EOL;
    }

?>
```

*Vi navngir denne filen som **for.php***

foreach

- Hvis du har en tydelig / bestemt struktur av data dvs det virker som en liste
- Tydelig start og slutt
- Traversere gjennom strukturen med foreach

```php
<?php

    $alleFrukt = array('Eple', 'Banan', 'Pære', 'Tomat');

    foreach ($alleFrukt as $frukt) {
       print $frukt . " er et frukt" . PHP_EOL;
    }

?>
```

*Vi navngir denne filen som **foreach.php***

for / while / foreach

- Når du har en bestemt struktur du ønsker å traversere og det er en tydelig start og slutt
  - ***foreach***
- Når du vet du skal gjøre noe et bestemt antall ganger
  - ***for***
- Når du skal gjøre noe flere ganger men slutt er usikker
  - ***while***
- Det vil ofte være det samme om du bruker en for / while

funksjoner

- Det er nyttig å avgrense programmet ditt i funksjoner
  - Lettere å gjenbruke koden din
  - Lettere å lese koden din

```php
<?php

    $alleFrukt = array('Eple', 'Banan', 'Pære', 'Tomat');

    foreach ($alleFrukt as $frukt) {
        printFrukt($frukt);
    }

    function printFrukt($frukt) {
        print $frukt . " er et frukt" . PHP_EOL;
    }

?>
```

*Vi navngir denne filen som **function.php***

Funksjon syntaks

```php
function funksjonsNavn () {

}

function funksjonsNavn ($variabel) {
    // tar imot en verdi
}

function funksjonsNavn () {
    // returner en verdi
   return $variabel;
}
```


PHP for eArkivarer

PHP og Databaser

Dag 1 Time 3

KDRS
Kommunearkivinstitusjonenes Digitale RessursSenter

Innhold

- PHP og databaser
- PHP og MySQL
- Hvordan lage en kobling til en database
  - Vi skal bare koble til og så lukke databasen
- Hvordan håndtere feil

PHP og databaser

- PHP kan koble seg opp mot de fleste DBHS
  - MySQL, Oracle, Postgres osv
- Avhengig at en bibliotek som tilbyr funksjonalitet er installert på maskinen din

PHP og MySQL

- Det er to mysql API (biblioteker) som kan brukes i PHP
  - mysql og mysqli
  - Bruk ***mysqli*** API
  - mysql ansees som foreldet

Koble til en database

```php
<?php

    $IPAdresse = ""; // Denne blir utdelt
    $databasenavn = "noark5";
    $brukernavn = "noarkBruker";
    $passord = "noarkPassord";

    $db = new mysqli($IPAdresse, $brukernavn, $passord, $databasenavn);

    print "Prøver å opprette kobling til " . PHP_EOL;
    print "maskin (" . $IPAdresse . "), " . PHP_EOL;
    print "database (" . $databasenavn . "), " . PHP_EOL;
    print "brukernavn (" . $brukernavn . "), " . PHP_EOL;
    print "passord (" . $passord . ")" . PHP_EOL;

    if ($db->connect_errno > 0){
        print "Kunne ikke koble til database (" . $db->connect_error . ")" . PHP_EOL;
    }
    else {
        print "Koblet til database" . PHP_EOL;
    }

    $db->close();

?>
```

*Vi navngir denne filen som **kobledatabase.php***

$db og ->

- $db variabelen er det vi kaller for en connector
  - Den gir oss en kobling til databasen
- Hva er $db?
  - $db = new mysqli(.....);
  - mysqli tilbyr en haug med funksjoner som vil kan bruke:
    - $db->query():
    - $db->close():

Håndtering av feil

- De vanligste feilene er:
  - Kan ikke koble til maskinen (IP-adressen)
    - **Can't connect to MySQL server on 'x.x.x.x' (110)**
  - MySQL-bruker har ikke lov til å koble til MySQL
    - **Access denied for user 'noarkBruker'@'localhost' (using password: YES)**
    - **grant SELECT on noark5.* to noarkBruker@localhost identified by 'noarkPassord';**
  - MySQL-bruker har ikke lov til å bruke den angitte databasen

Håndtering av feil

- Du bør alltid prøve å koble til MySQL fra kommandolinjen for å teste
  - **mysql -user=noarkBruker -password=noarkPassord**
  - Hvis du ikke kommer inn her da vil ikke PHP-skriptet ditt heller
- Da bør du prøve å bruke databasen du kobler deg opp mot
  - **use noark5;**
  - **show tables;**

PHP for eArkivarer

PHP og SQL

Dag 1 Time 4

KDRS
Kommunearkivinstitusjonenes Digitale RessursSenter

Innhold

- Utføre en spørring mot databasen
- Håndtere data
- Vi skal fortsatt jobbe med samme PHP-fil (kobledatabase.php) og vi bygger videre på den

Hva vi skal gjøre nå

- Vi skal nå
  - Lage en spørring (SQL)
  - Utføre spørringen mot databasen
  - Finne ut hvor mange rader av informasjon det er
  - Vi skal ikke se på data i tabellen enda
    - Bare se hvordan vi går fram
  - Rydde opp etter oss

Utføre en spørring

```php
<?php

   …
   // Forsetter med eksempelet fra forrige time

   $sql = "SELECT * FROM fonds";
   $result = $db->query($sql);

   if(!$result){
      die("Det oppsto et problem med spørringen  (" . $sql . ")." .
          PHP_EOL . "Feilen er (" . $db->error . ")");
   }

   $numberRowsInResultSet = $result->num_rows;
   print "Antall rader som vi kan forvente er (" .
         $numberRowsInResultSet . ")" . PHP_EOL;

   // Viktig å rydde opp etter oss
   $result->free();
   $db->close();

?>
```

*Vi navngir denne filen som **sporring1.php***

Hva vi skal gjøre nå

- Vi skal nå
  - Lage en spørring (SQL)
  - Utføre spørringen mot databasen
  - Finne ut hvor mange rader av informasjon det er
  - Vise innholdet av spørringen på skjerm
  - Rydde opp etter oss

Utføre en spørring

```php
<?php
    …
    // Forsetter med eksempelet fra forrige time
    $sql = "SELECT * FROM fonds";
    $result = $db->query($sql);

    if(!$result){
        die("Det oppsto et problem med spørringen  (" . $sql . ")." .
            PHP_EOL . "Feilen er (" . $db->error . ")");
    }

    $numberRowsInResultSet = $result->num_rows;
    print "Antall rader som vi kan forvente er (" .
          $numberRowsInResultSet . ")" . PHP_EOL;

    while($row = $result->fetch_assoc()){
        print "Tittel er (" . $row['title'] .")" . PHP_EOL;
    }

    $result->free();
    $db->close();
?>
```

*Vi navngir denne filen som **sporring2.php***

Hva ser vi på?

- Det er viktig å holde orden på objektene vi jobber med
  - $db er en kobling til databasen
  - $result er en kobling til spørringen
  - $row er en kobling to hver rad (tuppel) av data i spørringen

En rad om gangen

- $row er en *associative array* eller en *assosiativ matrise*
- Består av en (nøkkel, verdi) par
  - Nøkkelen er navnet på kolonnen mens verdi er dataene hentet fra raden (tuppel)
  - ['description'] => Registeret er organisert på gårds- og bruksnummer
  - ['title'] => Utvalgsarkiv
- $row = $result->fetch_assoc()
  - hver rad ($row) settes sammen som en assosiativ matrise

Utføre en spørring

- Vi bruker alltid samme måte når vi skal hente data fra en tabell
  - Så lenge det er data (while) prosesser det

```php
<?php

   while($row = $result->fetch_assoc()){
      print "SystemId er (" . $row['system_id'] .")" . PHP_EOL;
      print "Tittel er (" . $row['title'] .")" . PHP_EOL;
      print "Beskrivelse er (" . $row['description'] .")" . PHP_EOL;
   }

?>
```

Hvis du ikke vet strukturen?

- ***print_r()*** er en nyttig funksjon hvis du ønsker å inspisere en variabel og se innholdet i en menneskelesbar format

```php
<?php
    …
    while($row = $result->fetch_assoc()){
        print_r($row);
    }
    …
?>
```


Utføre en spørring (arkiv)

```php
<?php
    …
    $sqlArkiv = "SELECT * FROM fonds";
    $resultArkiv = $db->query($sqlArkiv);

    if(!$resultArkiv){
        die("Det oppsto et problem med spørringen  (" . $sqlArkiv . ")." .
             PHP_EOL . "Feilen er (" . $db->error . ")");
    }

    $numberArkivRows = $resultArkiv->num_rows;
    print "Antall arkiv som vi kan forvente er (" .
            $numberArkivRows . ")" . PHP_EOL;

    while($rowArkiv = $resultArkiv->fetch_assoc()){
        print "SystemId er (" . $rowArkiv['system_id'] .")" . PHP_EOL;
        print "Tittel er (" . $rowArkiv['title'] .")" . PHP_EOL;
        print "Beskrivelse er (" . $rowArkiv['description'] .")" . PHP_EOL;
    }

    $resultArkiv->free();
    $db->close();
?>
```

*Vi navngir denne filen som **sporring3.php***

Hva vi skal gjøre nå

- Lage to spørringer, en som går på arkiv (fonds) og en som går på arkivdel (series)
- Vise alle *arkiv* og alle *arkivdeler* som er koblet til de forskjellige *arkiv*
- Nå må gi mer fornuftige navn til:
  - Spørringen, $sql
    - $sqlArkiv og $sqlArkivdel
  - Resultatsettet, $result
    - $resultArkiv og $resultArkivdel
  - Radene, $row
    - $rowArkiv og $rowArkivdel

Utføre en spørring (arkiv/arkivdel)

```php
<?php
    …

    while($rowArkiv = $resultArkiv->fetch_assoc()){
        print "(arkiv) SystemId er (" . $rowArkiv['system_id'] .")" . PHP_EOL;

        $sqlArkivdel = "SELECT * FROM series where series_fonds_id = '" .
                       $rowArkiv['pk_fonds_id'] . "'";
        $resultArkivdel = $db->query($sqlArkivdel);

        if(!$resultArkivdel){
            die("\tDet oppsto et problem med spørringen  (" . $sqlArkivdel . ")." .
                PHP_EOL . "Feilen er (" . $db->error . ")");
        }

        $numberArkivdelRows = $resultArkivdel->num_rows;
        print "\tAntall arkivdel som vi kan forvente er (" .
                $numberArkivdelRows . ")" . PHP_EOL;

        while($rowArkivdel = $resultArkivdel->fetch_assoc()){
            print "\t(arkivdel) SystemId er (" . $rowArkivdel['system_id'] .")" . PHP_EOL;
            print "\t(arkivdel) Tittel er (" . $rowArkivdel['title'] .")" . PHP_EOL;
            print "\t(arkivdel) Beskrivelse er (" . $rowArkivdel['description'] .")" . PHP_EOL;
        }
        $resultArkivdel->free();
    }
    …
?>
```

*Vi navngir denne filen som **sporring4.php***

Noen tanker

- Nå begynner det å bli litt komplisert
  - Ikke lett å holde orden på strukturen
  1. Allikevel er det den samme grunnstrukturen som gjelder
     1. $sql = "select * from NOE" ;
     2. $result = $db->query($sql);
     3. if(!$result)
     4. while($row = $result->fetch_assoc())
     5. $result->free();

Oppgave

- Nå skal vise nivåene
  - Arkiv
  - Arkivdel
  - Saksmappe
- Det blir bare enda runde av den grunnlegende strukturen
  - Jobber utifra while-løkken til arkivdel
  - *Vi navngir denne filen som* ***sporring5.php***

PHP for eArkivarer

PHP og SQL til XML

Dag 1 Time 5 og 6

Innhold

- Konverte data til XML
- Skrive XML-data til en fil
- 'Rense' innhold

Hva vi skal gjøre nå

- Vi skal nå
  - Fortsette med PHP-fil (sporring3.php)
  - Markere data som XML

Innhold som XML

```
<?php
    …
    if ($numberArkivRows > 0) {
        print "<uttrekk>" .  PHP_EOL;

        while($rowArkiv = $resultArkiv->fetch_assoc()){
            print "\t<arkiv>" .  PHP_EOL;

            print "\t\t<systemId>" .
                    $rowArkiv['system_id'] . "</systemId>" . PHP_EOL;

            print "\t\t<tittel>" .
                    $rowArkiv['title'] ."</tittel>" . PHP_EOL;

            print "\t\t<beskrivelse>" .
                    $rowArkiv['description'] ."</beskrivelse>" . PHP_EOL;

            print "\t</arkiv>" . PHP_EOL;
        }
        print "</uttrekk>" . PHP_EOL;
    }
    …
?>
```

*Vi navngir denne filen som **db2xml.php***

Hva ser vi på?

- Det eneste vi har endret er at vi har innlemmet data i XML
  - Vi har egentlig ikke gjort noe annet, kun endret hvordan det ser ut
- Neste steg er å skrive det til en fil istedet for skjerm

Filhåndtering i PHP

- Det er enkelt å jobbe med filer i PHP
  - $filkobling = fopen(filnavn, modus)
    - Åpne en fil
    - Modus (lese fra, skrive til)
  - fwrite($filkobling, $data)
    - Skrive noe til en fil
  - fflush($filkobling)
    - Tving skrivingen
  - fclose($filkobling)
    - Lukk filen

Skrive XML til en fil

```php
<?php
    …
    $filnavn = "uttrekk.xml";
    $arkivFil = fopen($filnavn, "w");

    $numberArkivRows = $resultArkiv->num_rows;
    print "Antall arkiv som vi kan forvente er (" .
              $numberArkivRows . ")" . PHP_EOL;

    if ($numberArkivRows > 0) {
         fwrite($arkivFil, '<?xml version="1.0" encoding="UTF-8"?>' . PHP_EOL);
         fwrite($arkivFil, "<uttrekk>" .  PHP_EOL);

         while($rowArkiv = $resultArkiv->fetch_assoc()){
              fwrite($arkivFil, "\t<arkiv>" . PHP_EOL);
              fwrite($arkivFil, "\t\t<systemId>" . $rowArkiv['system_id'] . "</systemId>" . PHP_EOL);
              fwrite($arkivFil, "\t\t<tittel>" . $rowArkiv['title'] ."</tittel>" . PHP_EOL);
              fwrite($arkivFil, "\t\t<beskrivelse>" . $rowArkiv['description'] ."</beskrivelse>" . PHP_EOL);
              fwrite($arkivFil, "\t</arkiv>" . PHP_EOL);
         }
         fwrite($arkivFil, "</uttrekk>" . PHP_EOL);
    }
    fflush($arkivFil);
    fclose($arkivFil);

    …
?>
```

*Vi navngir denne filen som **db2xmlfil.php***

Igjen kan det virke komplisert

- Men det er samme grunnlegende struktur som tidligere
  - Bare vi skriver innholdet til en fil og ikke til skjerm
- Utgangspunktet er egentlig enkelt

'Rensing' av innhold

- Følgende tegn er problematisk i XML
  - < erstattes med <
  - & erstattes med &
  - > erstattes med >
  - " erstattes med "
  - ' erstattes med '
- "Obama" <obama@whitehouse.gov>
- "Obama" <obama@whitehouse.gov>

Hvordan rense innhold

- Det enkleste er å bruke PHP sin egne funksjoner som gjør dette
- Det er mange forskjellige løsninger

```php
<?php
    $doc = new DOMDocument();
    $fragment = $doc->createDocumentFragment();

    $data = "'Obama' <obama@whitehouse.gov>";
    $fragment->appendChild($doc->createTextNode($data));
    print $doc->saveXML($fragment);
?>
```

Oppgave

- Nå skal vi skrive nivåene
  - Arkiv
  - Arkivdel
- som XML til en fil

```
<?xml version="1.0" encoding="UTF-8"?>
<arkiv>
 …
    <arkivdel>
    …
    </arkivdel>
</arkiv>
```


Oppsummering av dagen

- Vi har lært
  - Grunnelgende skripting med PHP
    - Variabler og kontroll av logikk (if og while)
  - Hvordan vi lager en kobling til en database
  - Hvordan vi utfører en spørring og håndterer data
  - Hvordan data fra en tabell kan markeres i XML
  - Hvordan data fra en tabell kan markeres i XML og skrives til en fil

Dag 2

- Vi bygger videre på dagens arbeid med å skrive ut data til en XML-fil
  - Vi har «hacket» en løsning som skriver XML til en fil
  - I morgen skal vi gjøre dette «*ordentlig*»
  - Fortsetter med nivået <arkivdel> og <saksmappe>
- Så skal vi prosessere en XML-fil og skrive innholdet til en tabell
- Bruke det vi har lært og se hvordan vi kan utvikle et skript som kan lage en ADDML-fil

PHP for eArkivarer

PHP og XML

Dag 2 Time 1 til 3

KDRS
Kommunearkivinstitusjonenes Digitale RessursSenter

Innhold

- Skrive data til en XML-fil
- Lese data fra en XML-fil
- Forstå SAX og DOM tilnærminger og håndtering av XML-filer

SAX og DOM

- SAX og DOM er to forskjellige tilnærminger til å jobbe med XML-filer
- **DOM** står for **D**ocument **O**bject **M**odel og laster hele XML-filen i RAM
  - Får fort problemer med store filer
  - Men veldig rask
- **SAX** står for **S**imple **A**PI for **X**ML Parsing og laster kun deler av XML-filen i RAM
  - Lett å jobbe med store filer
  - Saktere enn DOM

Vi skal lage følgende XML-fil

```
<?xml version="1.0" encoding="UTF-8"?>
<uttrekk>
  <arkiv>
    <systemID>1</systemID>
    <beskrivelse>En beskrivelse</beskrivelse>
  </arkiv>
</uttrekk>
```


Lag en XML-fil med DOM-metoden

```
<?php

    $dom = new DOMDocument("1.0", "UTF-8");

    // lager <uttrekk> ... </uttrekk>
    $uttrekkRoot=$dom->createElement("uttrekk");
    $dom->appendChild($uttrekkRoot);

    // lager <arkiv> ... </arkiv>
    $arkivElement = $dom->createElement("arkiv");
    $uttrekkRoot->appendChild($arkivElement);

    // lager <systemID>1</systemID>
    $systemIDElement = $dom->createElement("systemID");
    $systemIDText = $dom->createTextNode("1");
    $systemIDElement->appendChild($systemIDText);
    $arkivElement->appendChild($systemIDElement);

    // lager <beskrivelse>En beskrivelse</beskrivelse>
    $beskrivelseElement = $dom->createElement("beskrivelse");
    $beskrivelseText = $dom->createTextNode("En beskrivelse");
    $beskrivelseElement->appendChild($beskrivelseText);
    $arkivElement->appendChild($beskrivelseElement);

    $dom->formatOutput = true;
    $dom->save("uttrekk.xml");
?>
```

*Vi navngir denne filen som **xmlDOM.php***

Lag en XML-fil med SAX-metoden

```php
<?php

    $sax = new XMLWriter();
    $sax->openURI('uttrekk.xml');
    $sax->startDocument('1.0','UTF-8');
    $sax->setIndent(true);

    $sax->startElement('uttrekk');

        $sax->startElement('arkiv');

            $sax->writeElement('systemID', '1');
            $sax->writeElement('beskrivelse', 'En beskrivelse');

        $sax->endElement();
    $sax->endElement();
    $sax->endDocument();

    $sax->flush();

?>
```

*Vi navngir denne filen som **xmlSAX.php***

SAX kontra DOM

- SAX-metoden virker enklere å lese og forstå sammenlignet med DOM-metoden
- Hvis du må manipulere XML-treet ditt før du skriver det ut til fil så trenger du DOM
  - Feks antall journalposter koblet til sak
- Flere API'er i PHP som kan brukes til å håndtere XML-filer
  - DOM
    - DOMDocument og SimpleXML
  - SAX
    - XMLReader og XMLWriter
  - XML Parser

Oppgave

- Nå skal vi lage en XML-fil fra innholdet i arkiv tabellen
- Vi bruker filen ***sporring3.php*** som utgangspunkt

Fra SQL til XML

```php
<?php
    …
    $addmlSAX = new XMLWriter();
    $addmlSAX->openURI('uttrekk.xml');
    //$addmlSAX->openURI('php://output');
    $addmlSAX->startDocument('1.0','UTF-8');
    $addmlSAX->setIndent(true);

    $addmlSAX->startElement('uttrekk');
        while($rowArkiv = $resultArkiv->fetch_assoc()){
            $addmlSAX->startElement('arkiv');
                $addmlSAX->writeElement('systemID', $rowArkiv['system_id']);
                $addmlSAX->writeElement('tittel', $rowArkiv['title']);
                $addmlSAX->writeElement('beskrivelse', $rowArkiv['description']);
                $addmlSAX->writeElement('arkivstatus', $rowArkiv['fonds_status']);
                $addmlSAX->writeElement('dokumentmedium', $rowArkiv['document_medium']);
                $addmlSAX->writeElement('opprettetDato', $rowArkiv['created_date']);
                $addmlSAX->writeElement('opprettetAv', $rowArkiv['created_by']);
                $addmlSAX->writeElement('avsluttetDato', $rowArkiv['finalised_date']);
                $addmlSAX->writeElement('avsluttetAv', $rowArkiv['finalised_by']);
            $addmlSAX->endElement();  // end arkiv
        }
    $addmlSAX->endElement();  // end uttrekk
    $addmlSAX->endDocument();
    $addmlSAX->flush();
    …
?>
```

*Vi navngir denne filen som **sql2xmlArkiv.php***

Lese XML-filer

- Nå skal vi laste inn og prosessere innholdet fra en XML-fil
  - DOM og SAX metoder
  - DOM
    - DOMDocument og SimpleXML
  - SAX
    - XMLReader
- Nå bruker vi arkivstruktur.xml og skal vise inneholder i alle saksmapper på skjerm

Lese en XML-fil med DOM-metoden

```php
<?php

   $xml = simplexml_load_file("arkivstruktur.xml");

   foreach($xml->arkiv->arkivdel->mappe as $mappe) {
      print "sytemID er (" . $mappe->systemID . ")" . PHP_EOL;
      print "tittel er (" . $mappe->tittel  . ")" . PHP_EOL;
   }

?>
```

*Vi navngir denne filen som **simpleXMLRead.php***

SimpleXML

- Det er veldig lett å jobbe med SimpleXML, men husk at hele filen lastes i minne
- SimpleXML støtter også XPath

```php
<?php
    $xml = simplexml_load_file("arkivstruktur.xml");
    $alleMapper = $xml->xpath('/arkiv/arkiv/arkivdel/mappe');

    foreach($alleMapper as $mappe) {
        print "sytemID er (" . $mappe->systemID . ")" . PHP_EOL;
        print "tittel er (" . $mappe->tittel  . ")" . PHP_EOL;
    }
?>
```


DOMDocument

```php
<?php

    $dom = new DOMDocument();
    $dom->load('arkivstruktur.xml');

    $alleMapper = $dom->getElementsByTagName("mappe");

    foreach($alleMapper as $mappe) {

        $systemIDElement = $mappe->getElementsByTagName( "systemID" );
        $systemID = $systemIDElement->item(0)->nodeValue;
        print "systemID er (" . $systemID . ")" . PHP_EOL;

        $tittelElement = $mappe->getElementsByTagName( "tittel" );
        $tittel = $tittelElement->item(0)->nodeValue;
        print "tittel er (" . $tittel  . ")" . PHP_EOL;
    }

?>
```

*Vi navngir denne filen som **domDocumentRead.php***

Lese en XML-fil med SAX-metoden

```php
<?php

    $xml = new XMLReader();
    $xml->open('arkivstruktur.xml', 'UTF-8');

     while($xml->read() && $xml->name !== 'mappe')
        ; // gjør ingenting

     while($xml->name === 'mappe' && $xml->nodeType == XMLReader::ELEMENT) {
        $mappe = new SimpleXMLElement($xml->readOuterXML());

        $systemID = $mappe['systemID'];
        print "systemID er (" . $systemID . ")" . PHP_EOL;

        $tittel = $mappe['tittel'];
        print "tittel er (" . $tittel . ")" . PHP_EOL;

        $xml->next('mappe');
    }

?>
```

*Vi navngir denne filen som **saxRead.php***

SAX - XMLReader

- XMLReader er komplisert og vanskelig å jobbe med
- Du må kontrollere mye og kode mye for å holde orden
  - Vanskelig hvis strukturen til XML-filen kan endre seg
- DOM metoden laster hele XML-filen i et trestruktur som du kan bruke
  - SAX går gjennom filen node for node
- Men det bruker lite minne

Noen tanker

- Jeg tror ikke dere skal manipulere innhold i XML-dokumenter
- Dere vi jobbe med satsvis konvertering / prosessering
- Men det kan være oppgaver som å fikse datoformat eller endre statusverdier
  - Men det vil også være mer satsvis konvertering
- Det er ingen grunn til at dere ikke skal ha en maskin med 16GB RAM for å håndtere XML-filer
  - SAX kontra DOM da?

PHP for eArkivarer

Fra XML til SQL

Dag 2 Time 4 og 5

KDRS
Kommunearkivinstitusjonenes Digitale RessursSenter

Innhold

- Lese data fra en XML-fil og skrive det til en database
- Bruker SimpleXML for enkelthetsskyld

Lese en XML-fil med DOM-metoden

- Lett å traverse strukturen
  - `$xml->arkiv->arkivdel->mappe`
- *Husk hele filen lastes i minne*

```php
<?php

   $xml = simplexml_load_file("arkivstruktur.xml");

   foreach($xml->arkiv->arkivdel->mappe as $mappe) {
      print "sytemID er (" . $mappe->systemID . ")" . PHP_EOL;
      print "tittel er (" . $mappe->tittel  . ")" . PHP_EOL;
   }

?>
```

INSERT INTO TABELL (attributtliste) VALUES (verdiliste);

INSERT arkiv

```
INSERT INTO arkiv (
    systemId, tittel,
    beskrivelse, arkivstatus, dokumentmedium,
    opprettetDato, opprettetAv)
VALUES
    (
    "b1b6b3f1-9186-4899-b3d9-4ebf0174727c", "KDRS Arkivet",
    "Test arkiv for kurs", "Opprettet", "Elektronisk arkiv",
    "2014-03-13", "admin"
    );
```

SQL

```php
<?php

    $IPAdresse = ""; // Denne blir utdelt
    $databasenavn = "noark5BrukerX"; // X blir utdelt
    $brukernavn = "noarkBruker";
    $passord = "noarkPassord";

    $db = new mysqli($IPAdresse, $brukernavn, $passord, $databasenavn);

    if($db->connect_errno > 0){
        print "Kunne ikke koble til database (" . $db->connect_error . ")" . PHP_EOL;
    }
    else {
        print "Koblet til database" . PHP_EOL;
    }

    $sql = "SELECT * FROM fonds";
    $result = $db->query($sql);

    if(!$result){
        die("Det oppsto et problem med spørringen  (" . $sql . ")." .
             PHP_EOL . "Feilen er (" . $db->error . ")");
    }

    …

    $result->free();
    $db->close();
?>
```


XML til SQL

*Vi bruker filen **simpleXMLRead.php*** som grunnlag for denne

```php
<?php
    …
    $xml = simplexml_load_file("arkivstruktur.xml");

    foreach($xml->arkiv as $arkiv) {

        $sqlInsert = "INSERT INTO fonds (system_id, title, description, fonds_status, " ;
        $sqlInsert = $sqlInsert . "document_medium, created_date, created_by, finalised_date," ;
        $sqlInsert = $sqlInsert . "finalised_by) VALUES (";
        $sqlInsert = $sqlInsert . "'" . $arkiv->systemID . "', ";
        $sqlInsert = $sqlInsert . "'" . $arkiv->tittel . "', ";
        $sqlInsert = $sqlInsert . "'" . $arkiv->beskrivelse . "', ";
        $sqlInsert = $sqlInsert . "'" . $arkiv->arkivstatus . "', ";
        $sqlInsert = $sqlInsert . "'" . $arkiv->dokumentmedium . "', ";
        $sqlInsert = $sqlInsert . "'" . $arkiv->opprettetDato . "', ";
        $sqlInsert = $sqlInsert . "'" . $arkiv->opprettetAv . "', ";
        $sqlInsert = $sqlInsert . "'" . $arkiv->avsluttetDato . "', ";
        $sqlInsert = $sqlInsert . "'" . $arkiv->avsluttetAv . "'";
        $sqlInsert = $sqlInsert . ")";

        print "INSERT blir (" . $sqlInsert . ")" . PHP_EOL;
        $result = $db->query($sqlInsert);

        if(!$result){
            die("Det oppsto et problem med spørringen  (" . $sqlInsert . ")." .
                 PHP_EOL . "Feilen er (" . $db->error . ")");
        }
    }
    …
?>
```

*Vi navngir denne filen som **xml2sql.php***

Oppgave

- Utvikle et skript som kan sette innholdet fra både arkiv og arkivdel i en tabell
- Husk – hvis du prøver å legge de samme dataene til en tabell, så vil du få duplikater i primærnøkkelen

Noen tanker

- Det er lett å snekre sammen en løsning men ikke enkelt å bygge noe som er robust og gjenbrukbar
- Nå prøver vi oss på en løsning til å lage en ADDML-fil

PHP for eArkivarer

ADDML

Dag 2 Time 6

Innhold

- Lage en skript som lager en ADDML fil av innholdet i en tabell

Hva er ADDML?

RegistreringsNr	UnderstellsNr	Farge	Produsent	Modell
LH12984	10946534	Red	Volkswagen	Golf
DK23491	9648573	Blue	Toyota	Yaris
BP12349	5523840	Green	Skoda	Fabia
ZT97495	2643923	White	Seat	Leon

```
LH12984;10946534;Red;Volkswagen;Golf
DK23491;9648573;Blue;Toyota;Yaris
```

datafil

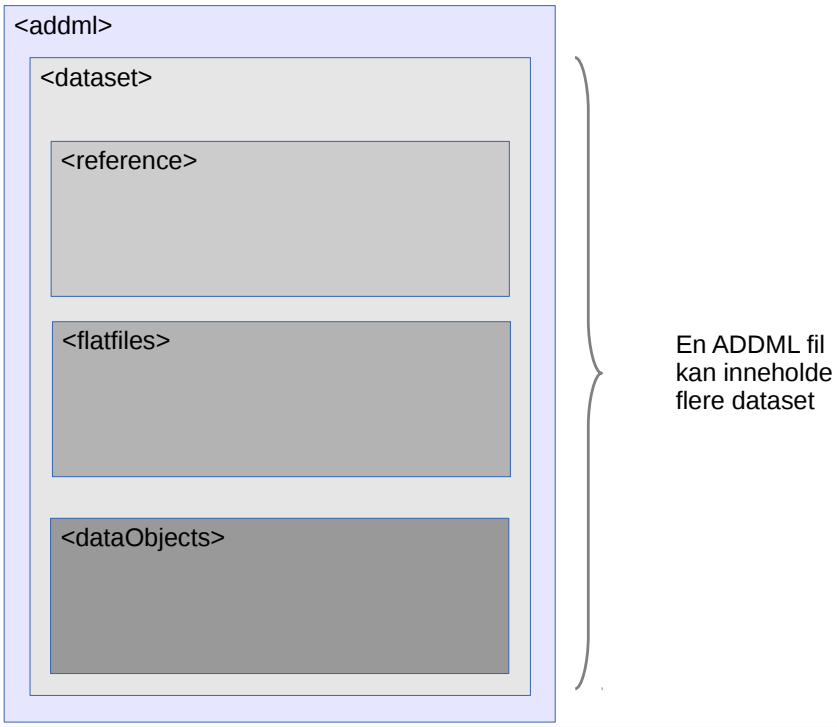
```
<addml>
…
RegistreringsNr
UnderstellsNr
Farge
Produsent
Modell
…
</addml>
```

tabell metadata

Tabelluttrekk beskrevet i ADDML

- ADDML kan brukes til å beskrive data
  - Typisk brukt til å beskrive innhold i tabeller fra relasjonsdatabase
  - Dataene er trukket ut i et eget datafil og ADDML-filen beskriver datafilen

Overordnet ADDML-fil



Overordnet ADDML-fil (PHP)

```php
<?php
    $addmlSAX = new XMLWriter();

    $addmlSAX->openURI('php://output');
    $addmlSAX->startDocument('1.0','UTF-8');
    $addmlSAX->setIndent(true);

    $addmlSAX->startElement('addml');
        $addmlSAX->writeAttribute('xmlns:xsi',
                    'http://www.w3.org/2001/XMLSchema-instance');
        $addmlSAX->writeAttribute('xmlns',
                    'http://www.arkivverket.no/standarder/addml');
        $addmlSAX->writeAttribute('xsi:schemaLocation',
                    'http://www.arkivverket.no/standarder/addml addml.xsd');

        $addmlSAX->startElement('dataset');
            $addmlSAX->startElement('flatFiles');
                …
            $addmlSAX->endElement();  // end flatfiles
        $addmlSAX->endElement();  // end dataset
    $addmlSAX->endElement();  // end addml

?>
```

Hva trenger vi ellers

- Kobling til en database
- Liste over
  - Primærnøkler
  - Fremmednøkler
  - Datatyper
  - Arkivskaper

Primærnøkler

- MySQL lar deg lett hente ut primærnøkler

```php
<?php

    function getPrimaryKeys($db, $tableName) {
        $query = "SHOW KEYS FROM " . $tableName . " WHERE Key_name = 'PRIMARY'";
        $primaryKeys = array();

        if($result = $db->query($query)){
            $i = 0;
            while($row = $result->fetch_assoc()){
                $primaryKeys[$i++] = $row['Column_name'];
            }
        }
        return $primaryKeys;
    }

?>
```

Fremmednøkler

- Litt vanskeligere å hente fremmednøkler

```php
<?php

     function getForeignKeys($db, $tableName) {

          $query = "SELECT COLUMN_NAME, REFERENCED_TABLE_NAME, " ;
          $query = $query . "REFERENCED_COLUMN_NAME FROM information_schema.key_column_usage " ;
          $query = $query . "WHERE table_name = '" . $tableName;
          $query = $query . "' AND CONSTRAINT_NAME != 'PRIMARY'";

          $foreignKeys = array();
          if($result = $db->query($query)){
               $i = 0;
               while($row = $result->fetch_assoc()){
                    $foreignKeys[$i++] = array ('fromCol' => $row['COLUMN_NAME'],
                              'toCol' => $row['REFERENCED_COLUMN_NAME'],
                              'toTable' => $row['REFERENCED_TABLE_NAME']);
               }
          }
          return $foreignKeys;
     }
?>
```


Datatyper

- Datatyper kan hentes med en spørring
  - Men også med en DESCRIBE eller SHOW COLUMNS kommando

```php
<?php

    function getColumnNameAndType($db, $tableName) {
        $query = "SELECT * FROM " . $tableName;
        $columnNameAndTypes = array();

        if($result = $db->query($query)){
            $i = 0;
            while ($columnInfo = $result->fetch_field()){
                $columnNameAndTypes[$i++] = array('name' => $columnInfo->name,
                    'type' => findTypeFromValue($columnInfo->type));
            }
        }
        return $columnNameAndTypes;
    }

?>
```

<context>

<addml>

<dataset>

```
<reference>
    <context>
     <additionalElements>
      <additionalElement name="recordCreators">
       <additionalElements>
        <additionalElement name="recordCreator">
         <value>KDRS</value>
        </additionalElement>
       </additionalElements>
      </additionalElement>
     </additionalElements>
     </context>
</reference>
```

<content>

<addml>

<dataset>

```
<reference>
     <content>
         <additionalElements>
               <additionalElement name="archivalPeriod">
                    <additionalElements>
                         <additionalElement name="startDate">
                              <value>20060401</value>
                         </additionalElement>
                         <additionalElement name="endDate">
                              <value>20100401</value>
                         </additionalElement>
                         <additionalElement name="period">
                            <additionalElements>
                                  <additionalElement name="inngåendeSkille">
                                       <value>Skarpt</value>
                                  </additionalElement>
                                  <additionalElement name="utgåendeSkille">
                                       <value>Skarpt</value>
                                  </additionalElement>
                             </additionalElements>
                         </additionalElement>
                    </additionalElements>
               </additionalElement>
          </additionalElements>
   </content>
</reference>
```

primærnøkkel

<addml>

<dataset>

```
<flatFiles>
    …
    <keys>
        <key name="primnokkel">
         <primaryKey/>
         <fieldDefinitionReferences>
          <fieldDefinitionReference name="pk_series_id"/>
         </fieldDefinitionReferences>
     </key>
    …
</flatFiles>
```

Håndtere primærnøkler

```
<?php

    $addmlSAX->startElement('keys');
        $addmlSAX->startElement('key');
            $addmlSAX->writeAttribute('name','primnokkel');
            $addmlSAX->startElement('primaryKey');
            $addmlSAX->endElement();  // end primaryKey

            $addmlSAX->startElement('fieldDefinitionReferences');
                foreach ($primaryKeys as $primaryKey) {
                    $addmlSAX->startElement('fieldDefinitionReference');
                        $addmlSAX->writeAttribute('name', $primaryKey);
                    $addmlSAX->endElement();  // end fieldDefinitionReference
                }
            $addmlSAX->endElement();  // end fieldDefinitionReferences
        $addmlSAX->endElement();  // end key

?>
```

kolonnebeskrivelse

<addml>

<dataset>

```
<flatFiles>
     …
      <fieldDefinitions>
        <fieldDefinition name="pk_series_id" typeReference="BIGINT"/>
        <fieldDefinition name="created_by" typeReference="VARCHAR"/>
        <fieldDefinition name="created_date" typeReference="DATETIME"/>
        <fieldDefinition name="description" typeReference="VARCHAR"/>
        <fieldDefinition name="document_medium" typeReference="VARCHAR"/>
        <fieldDefinition name="finalised_by" typeReference="VARCHAR"/>
        <fieldDefinition name="finalised_date" typeReference="DATETIME"/>
        <fieldDefinition name="series_end_date" typeReference="DATE"/>
        <fieldDefinition name="series_start_date" typeReference="DATE"/>
        <fieldDefinition name="series_status" typeReference="VARCHAR"/>
        <fieldDefinition name="system_id" typeReference="VARCHAR"/>
        <fieldDefinition name="title" typeReference="VARCHAR"/>
        <fieldDefinition name="series_classification_system_id"
                         typeReference="BIGINT"/>
        <fieldDefinition name="series_fonds_id" typeReference="BIGINT"/>
        <fieldDefinition name="referencePrecursor_pk_series_id"
                         typeReference="BIGINT"/>
      </fieldDefinitions>
     </recordDefinition>
     …
</flatFiles>
```

Håndtere kolonnebeskrivelser

```php
<?php
    …
    $addmlSAX->startElement('fieldDefinitions');
        foreach ($columnNameAndTypes as $columnNameAndType) {
            $addmlSAX->startElement('fieldDefinition');
                $addmlSAX->writeAttribute('name', $columnNameAndType['name']);
                $addmlSAX->writeAttribute('typeReference', $columnNameAndType['type']);
            $addmlSAX->endElement();  // end fieldDefinition
        }
    $addmlSAX->endElement();  // end fieldDefinitions
    …
?>
```

Noen tanker

- Dette vil bare fungere med MySQL-tabeller, mange systemer er i MSSQL, Oracle og andre DBHS
- Vi må enten bygge moduler som kan håndtere de forskjellige DBHS eller finne en API som gjør dette
- Se filen addmlGen.php