

Appendix B

til

Systemutvikling – fra kjernen og ut, fra skallet og inn

Webformularer, PHP og databaser

© Gerhard Skagestein, januar 2003
oppdatert 1. april 2003, 7. januar 2004

| | | |
|-------|--|------|
| B | Webformularer, PHP og databaser | B-3 |
| B.1 | Innledning | B-3 |
| B.2 | PHP | B-3 |
| B.2.1 | Hvordan virker PHP? | B-4 |
| B.2.2 | PHP i HTML eller HTML i PHP? | B-5 |
| B.2.3 | Et litt mer nyttig eksempel | B-6 |
| B.2.4 | Variable og konstanter | B-6 |
| B.2.5 | Arrayer | B-7 |
| B.2.6 | Praktisk feilfinningsutskrift | B-8 |
| B.2.7 | Kontrollstrukturer | B-9 |
| B.2.8 | Hente data fra brukeren | B-9 |
| B.3 | Webformularer | B-11 |
| B.3.1 | Webformular med GET | B-11 |
| B.3.2 | Webformular med POST | B-12 |
| B.3.3 | Method GET eller POST? | B-13 |
| B.3.4 | Mer avanserte formularer | B-14 |
| B.3.5 | Slå sammen HTML- og PHP-filen | B-16 |
| B.4 | Kobling mot database med SQL | B-18 |
| B.4.1 | Noen praktiske funksjoner | B-18 |
| B.4.2 | Autentisering | B-20 |
| B.4.3 | Et eksempelprogram – kjør SQL-spørringer | B-22 |
| B.5 | Et litt større system | B-24 |

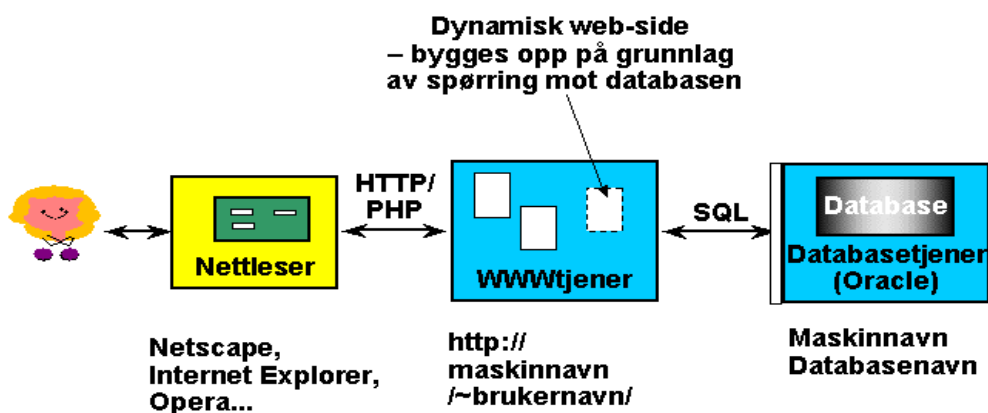
B Webformularer, PHP og databaser

Dette appendikset gir en innføring i bruk av webformularer og skriptspråket mot et databasehåndteringssystem. Det forutsettes at leseren kjenner til enkel HTML-koding, enkel programmering og litt om spørrespråket SQL.

Alle eksempelfilene finnes på filen <http://www.hoyskoleforlaget.no/kjernenskallet/php/phpfiler.zip>

Du bør opprette en mappe kalt php under www_docs på eget filområde og laste ned filene dit.

B.1 Innledning



Kombinasjonen av webformularer og PHP brukt mot et databasehåndteringssystem utgjør en interessant plattform for informasjonssystemer. Svært mange av dagens Internettbaserte systemer er da også bygd på denne eller liknende plattformer. Det er lett å forstå grunnen til at slike løsninger er blitt så populære:

- De utnytter standard Internett-teknologi
- De fleste brukere er kjent med nettlesere og hvordan de oppfører seg
- Det er enkelt å legge de enkelte deler av systemet på ulike maskiner som kan ha stor geografisk avstand – se figuren over.

Løsningen har imidlertid også en ulempe, nemlig at det er begrenset hvilken funksjonalitet vi kan få lagt inn i brukergrensesnittet. For å gjøre noe med det, må vi kjøre mer avanserte programmer på klienten enn en standard nettleser. Nærliggende løsninger er å bruke for eksempel Java-applets eller Java-scripts, men dette skal vi ikke komme inn på her.

B.2 PHP

Ved hjelp av skriptspråket PHP er det mulig å lage programmer som utføres på en webtjener. Det er et av de mest populære språkene for å lage dynamiske web-sider. PHP er "Open Source Software" – dvs. at det er fritt tilgjengelig på visse betingelser. PHP er en forkortelse for "PHP: Hypertext Preprocessor"¹. En fullstendig

¹ Tidligere "Personal Home Pages", men etter hvert er språket blitt utvidet til å kunne gjøre betydelig mer enn å gi websider en personlig vri.

dokumentasjon av språket kan finnes på PHPs offisielle nettsted på <http://www.php.net>.

Vår bruk av PHP skal begrense seg til å hente data som brukeren fyller ut på en webside, overføre data til og fra en database og dynamisk bygge opp en resultatside. I kombinasjon med webformularer og et databasehåndteringssystem er vi da i stand til å bygge enkle, men dog fullstendige informasjonssystemer på en enkel måte.

Mye kan sies om PHP som programmeringsspråk. Det har en Java-liknende syntaks og er meget uttrykkskraftig, og har et stort bibliotek av nyttige funksjoner. Det er imidlertid lite i språket som hindrer oss i å lage uryddige og uoversiktlige programmer. Språket har også utviklet seg over tid, og for at gamle programmer fremdeles skal fungere, aksepteres både foreldede og gjeldende språkkonstruksjoner. At det ikke er lett å følge med på hva som er gjeldende standard, kommer blant annet til uttrykk i at mange nye lærebøker om PHP bruker foreldede språkkonstruksjoner.

Programmene kompiles ikke på forhånd, og eventuelle syntaksfeil diagnostiseres derfor under kjøring med en relativt tynn feilmelding som for eksempel:

```
Parse error: parse error in  
/ifi/ganglot/a21/brukernavn/www_docs/php/test.php on line 4
```

B.2.1 Hvordan virker PHP?

Nedenfor er vist et eksempel på HTML-kode med innebygd PHP-kode. Denne teksten må være lagret i `www_docs` på en fil med filtype `php` – bruker vi filtype `htm` eller `html` som vanlig på websider, vil webtjeneren ignorere PHP-koden. Eksemplet ligger lagret på filen `hello.php`.

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

Det som skjer i web-tjeneren er at PHP-parseren² vil begynne å lese HTML-koden. Så lenge den ikke finner noe av interesse for seg, vil HTML-koden bli sendt uendret til nettleseren. Men så snart den ser `<?php` vil den begynne å interpretere den etterfølgende teksten som `php`-kommandoer og fortsette med dette helt til den støter på `?>`.

Her består PHP-programmet av en eneste programsetning, nemlig

```
print("Hello World<P>");
```

² Parser: Et program som leser og tolker en tekststreng

Når denne programsetningen utføres, vil teksten Hello Word <P> bli sendt over til nettleseren. Det betyr at idet webtjeneren når den avsluttende markeringen </html>, er følgende tekst sendt til nettleseren:

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

Dette er den teksten vi vil få se hvis vi på klientsiden ber om å få se HTML-kildeteksten for websiden ("Source"). PHP-koden er altså ikke synlig på klientsiden.

Bemerk at filen hello.php må ha UNIX adgangstillatelse 644 – dvs. rw-r--r--. Er dette ikke i orden, må du kjøre UNIX-kommandoen `chmod 644 hello.php`. Mapper med php-kode må ha adgangstillatelse 755 – dvs. rwxr-xr-x. Hvis adgangstillatelsene ikke er korrekte, får du en feilmelding a la

```
Warning: Failed opening
'/ifi/ganglot/a21/brukeravn/www_docs/php/hello.php' for
inclusion (include_path='.:www/httpd/apache_1.3.20_php-
4.0.6_mod_perl-1.25/lib/php') in Unknown on line 0
```

B.2.2 PHP i HTML eller HTML i PHP?

Eksemplet i forrige avsnitt kunne med samme virkning vært kodet slik (ligger på filen hello1.php):

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

Dette eksemplet illustrerer et poeng: Det er fullstendig likegyldig om teksten som sendes til nettleseren er selve HTML-koden eller om den "printes" av PHP. Faktisk kan hele HTML-teksten genereres av et PHP-program. Besvergelsen `\n` sørger for at det genereres et linjeskift.

B.2.3 Et litt mer nyttig eksempel

La oss se på et eksempel som er litt mer spennende enn å skrive ut en statisk tekst. Følgende kode som ligger på filen `phpdato.php` henter dagens dato fra webtjeneren, formaterer den til det lokale språk og sender den til nettleseren med litt pynt i form av en overskrift:

```
<html>
<head>
<title>PHP-dato</title>
</head>
<body>
<h1>Dagens dato er</h1>
    <?php
        setlocale(LC_ALL, 'no_NO'); // sett norsk språk
        print(strftime ("%A %d %B %Y ")); // hent tidspunkt
                                           // og formater
    ?>
</body>
</html>
```

Funksjonen `strftime` (`string format time`) henter opp dato og formaterer den i henhold til de noe kryptiske formateringsdirektivene `%A %d osv.` Dokumentasjon av både funksjonen og formateringsdirektivene finnes på <http://www.php.net>.

Merk at det tidspunktet som hentes, er tjenertid, ikke klienttid. Det kan være av en viss betydning hvis klient og tjener befinner seg i ulike tidssoner.

B.2.4 Variable og konstanter

Følgende kode på filen `sirkelomkretsfast.php` beregner omkretsen av en sirkel med radius 1 og legger resultatet på en webside som sendes til nettleseren:

```
<html>
<head>
    <title>PHP sirkelberegning</title>
</head>
<body>
    <H1>PHP sirkelberegning</h1>
    <?php
        define("PI", "3.1415926535897932");
        $radius = 1.0;
        print("Radius er ".$radius);
    ?>
    <P>
    <?php
        $omkrets = 2 * $radius * PI;
        print("Omkrets er ".$omkrets);
    ?>
</body>
</html>
```

Dette eksemplet viser følgende:

- Hvordan vi definerer konstanter
Dette gjøres ved hjelp av funksjonen define:
`define("PI", "3.1415926535897932");`
Det er vanlig å bruke store bokstaver for konstant-navn.
- Hvordan vi deklarerer en variabel og tilordner en verdi til den
Variabelnavn begynner alltid med tegnet \$. Variabelen opprettes første gang den nevnes. Tilordning skjer ved hjelp av tilordningsoperatoren =.
Eksempel: `$radius = 1.0;`
- Hvordan vi setter opp et aritmetisk uttrykk
Aritmetiske uttrykk settes opp og beregnes på tilsvarende måte som i andre programmeringsspråk.
Eksempel: `2 * $radius * PI`
- Hvordan vi skjøter sammen tekststrenger
Tekststrenger skjøtes sammen med operatoren . (punkt)

Tekststrenger (datatypen `string`) kan avgrenses både med apostrof ' og anførselstegn ". Forskjellen er at med apostrof brukes tekststrengen som den er, men en tekststreng i anførselstegn kan inneholde variabler hvis verdi vil bli satt inn i strengen, og konstruksjoner som `\n` blir konvertert til linjeskift. Dersom en tekststreng som inneholder et anførselstegn også avgrenses med anførselstegn, må vi skyte inn "escape-symbolet" `\` foran anførselstegnet i tekststrengen for at det ikke skal bli mistolket som avslutning på tekststrengen, slik som i dette eksemplet:

```
"PHP er en forkortelse for \"PHP Hypertext Processor\""
```

Tilsvarende gjelder selvsagt dersom vi har apostrofer i en tekststreng avgrenset med apostrofer.

PHP er et svakt typet språk, hvilket betyr at en og samme variabel etter hvert kan komme til å inneholde ulike typer data. Mulighetene er `integer`, `double`, `string` og også ikke-basis typer som `array` og objekter.

PHP-funksjonene er "case-insensitive" – `PRINT` gjør altså samme nytten som `print`. Derimot er variabel- og konstantnavn "case-sensitive" – `$PI` er altså ikke det samme som `$pi`.

B.2.5 Arrayer

I likhet med andre språk har PHP arrayer. Et array kan bl.a. opprettes ved hjelp av språkkonstruksjonen `array`:

```
$innbyggertall = array(243585, 453490, 499693);
```

```
$fylkenavn = array('Østfold', 'Akershus', 'Oslo');
```

Fordi PHP er svakt typet, kan et array inneholde en fager blanding av ulike typer data:

```
$innbyggertall = array(243585, 'ukjent', 499693);
```

Elementene indekseres fra 0, og kan hentes ut på samme måte som vi er vant til fra andre programmeringsspråk:

```
print($innbyggertall[0]); // skriver ut 243585
```

Spesielt for PHP er at arrayene også kan være assosiative, dvs. at hvert element har en nøkkel, og elementet kan hentes ut ved hjelp av denne nøkkelen. Koblingen mellom nøkkel og elementverdi angis med konstruksjonen `nøkkel => verdi`:

```
$innbyggertall = array('Østfold'=>243585,  
'Akershus'=>'ukjent', 'Oslo'=>499693);
```

Vi kan oppnå det samme med

```
$innbyggertall['Østfold'] = 243585;  
$innbyggertall['Akershus'] = 'ukjent';  
$innbyggertall['Oslo'] = 499693;
```

Et element kan nå ganske enkelt hentes ut ved å bruke en nøkkelverdi som indeks:

```
print($innbyggertall['Oslo']); // skriver ut 499693
```

Dersom vi henter opp en liten databasetabell med følgende utseende,

| FYLKENR | FYLKENAVN | INNBYGGERTALL |
|---------|-----------|---------------|
| 01 | Østfold | 243585 |
| 02 | Akershus | 453490 |
| 03 | Oslo | 499693 |

vil resultatet bli et assosiativt array med vanlige arrays som elementer, dvs samme struktur som vi vil få ved å utføre:

```
$fylkedata = array('FYLKENR'=>array('01', '02', '03'),  
                  'FYLKENAVN'=>array('Østfold', 'Akershus', 'Oslo'),  
                  'INNBYGGERTALL'=>array(243585, 453490, 499693));
```

B.2.6 Praktisk feilfinningsutskrift

For å finne feil i programmer kan det være til stor hjelp å få skrevet ut verdien av en variabel, uansett hvor sammensatt den er – for eksempel et array der elementene er dels nye arrayer, dels tall, dels tekststrenger. Funksjonen `var_dump($variabel)` ordner dette! Eksempelvis gir

```
var_dump($fylkedata);
```

brukt etter tilordningen til `$fylkedata` i avsnitt B.2.1 denne utskriften:

```
array(3) { ["FYLKENR"]=> array(3) { [0]=> string(2) "01" [1]=>  
string(2) "02" [2]=> string(2) "03" } ["FYLKENAVN"]=> array(3)  
{ [0]=> string(7) "Østfold" [1]=> string(8) "Akershus" [2]=>  
string(4) "Oslo" } ["INNBYGGERTALL"]=> array(3) { [0]=>  
int(243585) [1]=> int(453490) [2]=> int(499693) } }
```

B.2.7 Kontrollstrukturer

I likhet med andre programmeringsspråk omfatter også PHP valg- og løkkekonstruksjoner. Likheten med JAVA er definitivt tilstede:

IF-setninger

```
if (INSTITUTT == "Ifi") {
    print ("Institutt for informatikk");
}

if (INSTITUTT == "Ifi") {
    print ("Institutt for informatikk");
}else {
    print ("Ikke Institutt for informatikk");
}
```

FOR-løkker

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

WHILE-løkker

```
while ($antall < 15) {
    ...
    $antall = $antall + 1;
}
```

B.2.8 Hente data fra brukeren

I B.2.4 så vi på et eksempel der vi beregnet omkretsen av en sirkel med radius 1. Vi skal nå se på hvordan vi kan la brukeren gi radiusverdien. En måte å gjøre det på er å forlenge URLen med et spørsmålstegn etterfulgt av teksten radius=xxxx, der xxx er radiusverdien.

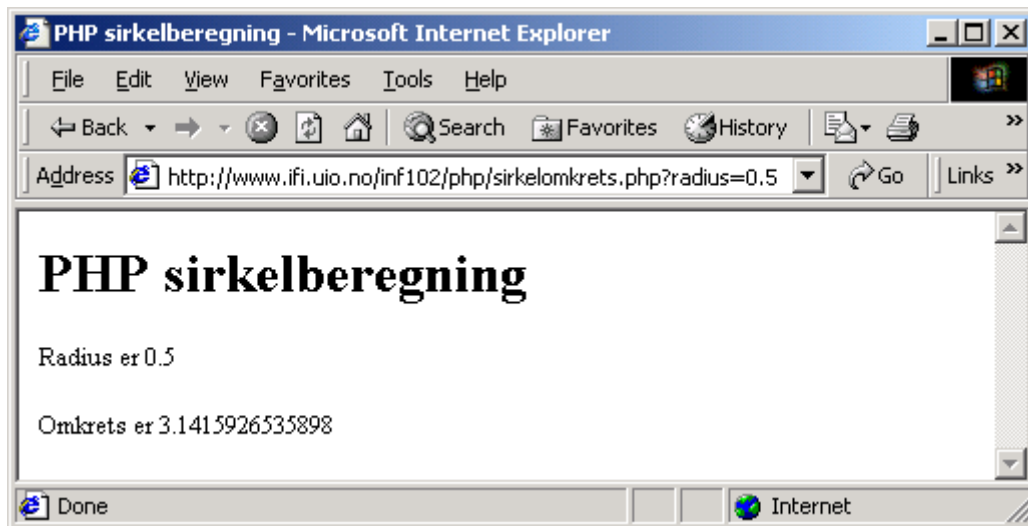
<http://www.hoyskoleforlaget.no/kjernenskallet/php/sirkelomkrets.php?radius=0.5>

Programmet ligger altså på sirkelomkrets.php:

```
<html>
<head>
    <title>PHP sirkelberegning</title>
</head>
<body>
    <H1>PHP sirkelberegning</h1>
    <?php
        define("PI", "3.1415926535897932");
        $radius = (double) $_GET['radius'];
        print("Radius er ".$radius);
    ?>
    <P>
    <?php
        $omkrets = 2 * $radius * PI;
        print("Omkrets er ".$omkrets);
    ?>
</body>
```

</html>

Det som skjer er at dataene etter spørsmålstegnet overføres til webtjeneren og legges i et assosiativt array ved navn `$_GET`. Derfra kan vi hente verdiene ved å indeksere dette arrayet med navnet på parameteren – her `radius`. Tilordningen `$radius = (double) $_GET['radius']` sørger for å konvertere tekststrengen fra URL'en til tallet 0.5 og lagre det i `$radius`. Deretter går PHP-programmet videre som i eksemplet i avsnitt B.2.4. Resultatet blir som følger:



B.3 Webformularer

Vi har nå sett hvordan vi kan få PHP-programmet til å beregne omkretsen av en sirkel med en radius som gis av brukeren ved å henge noen data på en URL. Imidlertid er det så som så med brukervennligheten. For å forbedre den må vi ta i bruk webformularer.

B.3.1 Webformular med GET

Her ser vi et eksempel på et enkelt webformular:



HTML-koden ligger på filen sirkelform.html og ser slik ut:

```
<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><INPUT TYPE="SUBMIT" VALUE="Beregn omkrets">
```

```
<INPUT TYPE="RESET" VALUE = "Tøm inndatafelt"></p>

</FORM>
<HR>
<I>Laget av NN dato</I>
</BODY>

</HTML>
```

Den interessante delen av HTML-koden befinner seg mellom markeringene `<FORM...>` og `</FORM>`. Det er denne delen av koden som gjør det mulig å opprette et formular med felter der brukeren kan legge inn data. Feltene har markeringene `<INPUT...>`. I `INPUT`-markeringen kan vi legge til noen viktige egenskaper:

- `NAME` = Et navn vi velger for feltet. Samme navn brukes som indeks i de assosiative arrayene `$_GET` og `$_POST` i programmet.
- `SIZE` = størrelsen på feltet på websiden
- `TYPE` = hva slags type felt – mer om dette siden!

Det finnes to spesielle `INPUT`-felt, nemlig med `TYPE= "SUBMIT"` og `TYPE="RESET"`. De resulterer i knapper som brukeren kan trykke på. `VALUE` angir teksten på knappen. Et trykk på `RESET`-knappen vil føre til at alle feltene i formularet tømmes. Et trykk på `SUBMIT`-knappen vil føre til at nettleseren kaller opp URLen som er oppført som egenskapen `ACTION` i `FORM`-markeringen. Samtidig vil dataene som brukeren har lagt inn i feltene følge med. Detaljene i hvordan dette gjøres, bestemmes av egenskapen `METHOD` i `FORM`-markeringen.

Så når brukeren i dette eksemplet trykker på knappen "Beregn omkrets", kalles `sirkelomkrets.php` – som jo er angitt i `ACTION` – automatisk opp, og `METHOD="GET"` sørger for at brukerens data følger med på nøyaktig samme måte som i forrige eksempel. Webtjeneren ser altså ingen forskjell på om dataene kommer fra et selvsnekret oppkall med data etter et spørsmålstegn eller om det kommer fra et formular. Det betyr at vi ikke behøver å gjøre noen som helst endringer på filen `sirkelomkrets.php`.

B.3.2 Webformular med POST

Ved å erstatte `METHOD="GET"` med `METHOD="POST"` blir dataene ikke overført som påheng på URL'en, men separat bak kulissene. I PHP-programmet er dataene nå å finne i det assosiative arrayet `$_POST`.

Koden for webformularet ligger på `sirkelformpost.html` og ser slik ut:

```
<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><INPUT TYPE="SUBMIT" VALUE="Beregn omkrets">
    <INPUT TYPE="RESET" VALUE = "Tøm inndatafelt"></p>
</FORM>
<HR>
<I>Laget av NN dato</I>
</BODY>
</HTML>
```

og filen sirkelomkretspost.php vil se slik ut:

```
<html>
<head>
    <title>PHP sirkelberegning</title>
</head>
<body>
    <H1>PHP sirkelberegning</H1>
    <?php
        define("PI", "3.1415926535897932");
        $radius = (double) $_POST['radius'];
        print("Radius er ".$radius);
    ?>
    <P>
    <?php
        $omkrets = 2 * $radius * PI;
        print("Omkrets er ".$omkrets);
    ?>
</body>
</html>
```

B.3.3 Method GET eller POST?

Sett fra brukeren side blir det liten forskjell om vi bruker METHOD="GET" eller METHOD="POST". Som programmerere bør vi imidlertid være klar over følgende:

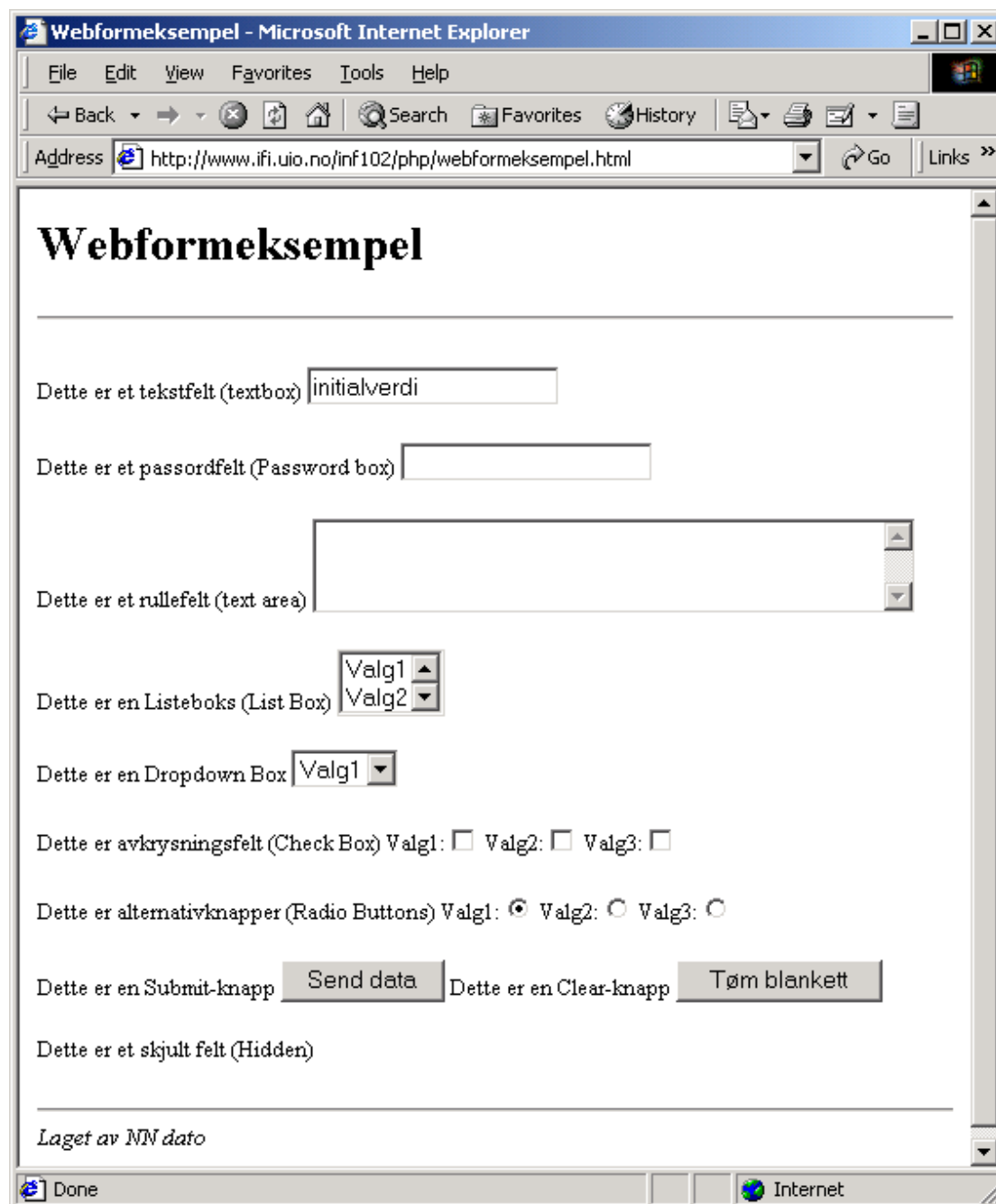
- GET blir brukt dersom METHOD ikke er angitt i FORM-markeringen.
- Med GET er dataene svært synlige og lette å stjele. GET bør derfor ikke brukes for sensitive data, og under ingen omstendigheter for brukernavn og passord.
- POST er å foretrekke for litt større datamengder.
- Dersom brukeren gjentar en forespørsel med samme URL og samme inndata, vil den utvidete URL være den samme som før, og ved bruk av GET-metoden vil nettleseren lete i webside-"cachen" etter svaret. Web-tjeneren får altså ikke vite noe om den gjentatte forespørselen, og har derfor ingen muligheter for å oppdatere svaret ut fra nye data på tjeneren. En forespørsel med POST-metoden, derimot, vil alltid bli bearbeidet av tjeneren.

Noen retningslinjer:

- Bruk GET for overføring av søkedata som brukes for å gjenfinne informasjon som ikke til stadighet endrer seg
- Bruk POST for sensitive data, for større datamengder, og for data som skal inn i en database.

B.3.4 Mer avanserte formularer

Ved hjelp av egenskapen TYPE i INPUT-markeringen eller ved hjelp av SELECT-markeringen kan vi sette opp en rekke forskjellige typer felter i formularet. Her følger en eksempelside som viser de vanligste mulighetene. I tillegg kan vi sette initialverdier i feltene med VALUE=" initialverdi"



HTML-koden ligger på webformeksempel.html og ser slik ut:

```
<HTML>
<HEAD>
<TITLE>Webformeksempel</TITLE>
</HEAD>
```

```

<BODY>
<H1>Webformeksempel</H1>
<HR>
<FORM METHOD = "POST" ACTION = "mailto:din_epostadresse">

<P>Dette er et tekstfelt (textbox) <INPUT TYPE="TEXT" SIZE =
"20" NAME="dintekst" VALUE="initialverdi">

<P>Dette er et passordfelt (Password box) <INPUT
TYPE="PASSWORD" SIZE = "20" NAME="dittpassord">

<P>Dette er et rullefelt (text area) <TEXTAREA NAME =
"dinlangetekst" ROWS = "3" COLS = "40" ></TEXTAREA>

<P>Dette er en Listeboks (List Box)
<SELECT NAME="listevalg" SIZE="2">
  <OPTION>Valg1</OPTION>
  <OPTION>Valg2</OPTION>
  <OPTION>Valg3</OPTION>
</SELECT>

<P>Dette er en Dropdown Box
<SELECT NAME="dropdownvalg">
  <OPTION VALUE="1">Valg1</OPTION>
  <OPTION VALUE="2">Valg2</OPTION>
  <OPTION VALUE="3">Valg3</OPTION>
</SELECT>

<P>Dette er avkrysningsfelt (Check Box)
  Valg1:<INPUT TYPE="CHECKBOX" NAME="Valg1">
  Valg2:<INPUT TYPE="CHECKBOX" NAME="Valg2">
  Valg3:<INPUT TYPE="CHECKBOX" NAME="Valg3">
</p>

<P>Dette er alternativknapper (Radio Buttons)
  Valg1:<INPUT TYPE="RADIO" NAME="radioknapp" VALUE = "Valg1"
CHECKED>
  Valg2:<INPUT TYPE="RADIO" NAME="radioknapp" VALUE = "Valg2">
  Valg3:<INPUT TYPE="RADIO" NAME="radioknapp" VALUE = "Valg3">

<P>Dette er en Submit-knapp <INPUT TYPE="SUBMIT" VALUE="Send
data">
Dette er en Clear-knapp <INPUT TYPE="RESET" VALUE = "Tøm
blankett"></P>

<P>Dette er et skjult felt (Hidden) <INPUT TYPE="HIDDEN"
NAME="skjultedata" VALUE = "jeg er usynlig"></P>

</FORM>
<HR><I>Laget av NN dato</I>
</BODY>

</HTML>

```

For enkelt å kunne se hvilke data som blir sendt når du fyller ut webformularet, kan du sette inn e-post-adressen din som verdi for ACTION i FORM-markeringen. Da vil du motta dataene som vedlegg til en e-post når du trykker på ”Send data”-knappen. Særlig dataene som sendes ved bruk av avkrysningsfelter og radioknapper kan være verd et studium.

B.3.5 Slå sammen HTML- og PHP-filen

I eksemplene så langt har vi arbeidet med en HTML-fil som setter opp formularet som brukeren skal fylle ut, og en PHP-fil som fanger opp dataene og gjør det som skal gjøres. Mange liker å slå disse to filene sammen til en eneste PHP-fil. Dette har den fordel at formularet kan brukes også for å vise fram resultatet av utførelsen av PHP-programmet. Dessuten reduserer det antall filer til det halve, og man er ikke i tvil om hvilke HTML-formularer og PHP-programmer som hører sammen.

Det er ikke store endringene som skal til: Vi lager en PHP-fil med PHP-koden først og HTML-formularet sist (motsatt rekkefølge går også, men logikken blir lettere å følge på denne måten). Deretter sørger vi for at egenskapen ACTION i FORM-markeringen kaller den selvsamme PHP-filen.

Imidlertid er det en komplikasjon: For å få fram formularet er vi nødt til å kalle PHP-filen, og da vil PHP-programmet første gang bli utført før brukeren har hatt mulighet til å fylle ut formularet. Derfor bør PHP-programmet ta høyde for dette, for eksempel ved å teste om elementet finnes i \$_GET eller \$_POST og om det finnes data i INPUT-feltene. Dette er for øvrig en sunn test også for senere kall på PHP-programmet, i tilfelle brukeren skulle finne på å trykke på SUBMIT-knappen uten å ha fylt ut feltene i formularet.

Som eksempel tar vi HTML-filen og PHP-programmet fra B.3.2 og slår dem sammen. Resultatet ligger på filen `sirkelberegning.php`:

```
<?php // ligger på sirkelberegning.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);?> >

    <P><INPUT TYPE="SUBMIT" VALUE="Beregn omkrets">
    <INPUT TYPE="RESET" VALUE = "Tøm inndatafelt"></p>
</FORM>
```

```
<P>
<?php print($resultat); ?>
<HR>
<I>Laget av NN dato</I>
</BODY>
</HTML>
```

Legg merke til de to små bitene med PHP-kode i HTML-delen. Den første sørger for å skrive inn radius-verdien som brukes som utgangspunkt for beregningen inn igjen i INPUT-feltet, slik at verdien her stemmer overens med resultatet som vises på samme skjermbilde. Innledningsvis settes radius-verdien til 0. Den andre sørger for å skrive ut resultatet på et hensiktsmessig sted, i dette tilfelle nedenfor SUBMIT-knappen.

Skjermbildet ser slik ut:



B.4 Kobling mot database med SQL

Det er blitt stadig vanligere å knytte web-dokumenter sammen med databaser, derfor tilbyr de fleste scriptspråk etter hvert god støtte for slike koblinger. PHP er intet unntak, og inneholder derfor en rekke innebygde funksjoner for å kommunisere med databasehåndteringssystemer som MySQL og Oracle. Eksempler på funksjoner i grensesnittet mot databasehåndteringssystemet Oracle er:

- OCILogon // for innlogging i basen
- OCIlogoff // for utlogging av basen
- OCIExecute // for å utføre en SQL-kommando

B.4.1 Noen praktiske funksjoner

For å gjøre det enklere å arbeide mot en Oracle-database er det laget noen funksjoner som ligger på filen `db.inc` i mappen `inc`. Her følger en beskrivelse.

```
$conn = baseLogon($bruker, $passord, $database)  
logger inn i databasen $database som brukeren $bruker med $passord.  
Funksjonen vil returnere en kobling mot databasen som her lagres i $conn.
```

```
baseLogoff($conn)  
vil stenge koblingen $conn og logge ut av databasen koblingen var satt opp mot.
```

```
$stmt = baseQuery($conn, $query)  
vil stille spørresetningen $query til databasen som er angitt med koblingen $conn.  
Returnerer et "håndtak" (peker) til et internt resultat av spørringen som her lagres i $stmt.  
Spørreresultatet må bearbeides videre (se nedenfor) for å bli leselig.
```

```
$pkattributter = finnPKAttributter($conn, $tabell)  
returnerer et array med navnene på de attributtene som inngår i primærnøkkelen i tabellen $tabell i henhold til metadatabasen med koblingen $conn. Her lagres pekeren til arrayet i $pkattributter
```

Eksempel: Hvis `FYLKENR` er definert som primærnøkkel i tabellen `FYLKE`, vil funksjonen returnere `array('FYLKENR')`.

```
$query = byggUpdateQuery($elements, $tabell, $pkattributter)  
returnerer et SQL UPDATE-query som i tabellen $tabell oppdaterer attributtene som er gitt i det assosiative arrayet $elements, der elementene har formen 'attributtnavn'=>attributtverdi. Attributter som også er gitt i arrayet $pkattributter oppdateres imidlertid ikke; verdiene for disse attributtene brukes istedenfor til å identifisere hvilke linjer som skal oppdateres. SQL-spørringen lagres her i $query.
```

Eksempel: Hvis

```
$elements = array('FYLKENR'=>'01', 'INNBYGGERTALL'=>25000);  
$tabell = 'FYLKE';  
$pkattributter = array('FYLKENR');  
vil funksjonen returnere  
UPDATE TABLE FYLKE SET INNBYGGERTALL=25000 WHERE FYLKENR='01'
```

`$query = byggDeleteQuery($elements, $tabell, $pkattributter)`
 returnerer et SQL DELETE-query som i tabellen `$tabell` fjerner de linjene der attributtene har verdier som angitt i det assosiative arrayet `$elements`, der elementene har formen 'attributtnavn'=>attributtverdi, og attributtene samtidig er gitt i arrayet `$pkattributter`. SQL-spørringen lagres her i `$query`.

Eksempel: Hvis

```
$elements = array('FYLKENR'=>'01');
$tabell = 'FYLKE';
$pkattributter = array('FYLKENR');
vil funksjonen returnere
DELETE FROM TABLE FYLKE WHERE FYLKENR='01'
```

`$query = byggInsertQuery($elements, $tabell)`
 returnerer et SQL INSERT-query som i tabellen `$tabell` setter inn nye linjer med attributtverdier som er gitt i det assosiative arrayet `$elements`, der elementene har formen 'attributtnavn'=>attributtverdi. SQL-spørringen lagres her i `$query`.

Eksempel: Hvis

```
$elements = array('FYLKENR'=>'21',
'FYLKENAVN'=>'Svalbard', 'INNBYGGERTALL'=>1151);
$tabell = 'FYLKE';
vil funksjonen returnere
INSERT INTO TABLE FYLKE (FYLKENR, FYLKENAVN, INNBYGGERTALL)
VALUES ('21','Svalbard', 1151)
```

Men vi skal ikke bare hente data fra databasen, vi skal også kunne vise fram dataene og kunne oppdatere databasen. For dette formål finnes det ytterligere noen hjelpefunksjoner som ligger på filen `gui.inc` i mappen `inc`. Her følger en beskrivelse. Alle funksjonene leser det interne resultatet av en foregående spørring pekt til fra variabelen `$stmt` og returnerer HTML-kode som i beskrivelsen nedenfor forutsettes lagret i variabelen `$html`.

`$html = lagSelectMeny($nokkel, $stmt)`
 returnerer HTML-kode for en nedtrekksmeny med verdien i `$nokkel` som navn på variabelen som vil bli sendt med ved en `SUBMIT`. Navnet må være identisk med et av attributtene vi valgte ut i den foregående spørringen, helst primærnøkkelen.

Eksempel:

`$html = lagSelectMeny("FYLKENR", $stmt)` vil returnere HTML-kode for bildet på side B-24. Etter at brukeren har valgt (02) Akershus og klikket "Velg Fylke", sendes `FYLKENR=02` videre som parameter.

`$html = visTabell($stmt, $pkattributter)`
 returnerer HTML-kode for å vise fram tabellen som ligger i spørreresultatet `$stmt`. Attributtnavn som er gitt i arrayet `$pkattributter` vil bli vist med rød bakgrunnsfarge.

`$html = visTabellMedLink($stmt, $pkattributter, $hreffil)`
 returnerer HTML-kode tilsvarende `visTabell`, men med en ekstra kolonne med en aktiv link til en web-side `$hreffil`. Attributtnavn i arrayet `$pkattributter` med tilhørende verdier blir hengt på som parametre på linken.

`$html = lagOppdateringsformular($stmt, $pkattributter)`
returnerer HTML-kode som lager et formular som gjør det mulig å endre, legge til, eller fjerne linjer i tabellen som ligger i spørreresultatet `$stmt`. Attributnavnene i arrayet `$pkattributter` spiller en avgjørende rolle ved oppdatering og fjerning av linjer, siden det er verdiene i disse attributtene som bestemmer hvilke linjer som skal berøres.

`$html = lagInnleggingsformular($stmt, $pkattributter)`
returnerer HTML-kode som lagOppdateringsformular, med den endringen at formularet har tomme felter og dermed er klart for å legge inn data for en ny linje i tabellen som ligger i bufferen `$stmt`.

B.4.2 Autentisering

Som kjent trengs det et brukernavn og passord for å kunne logge seg inn i databasen. PHP-programmet nedenfor som ligger på filen `autentiser.php` setter opp et HTML-formular der brukeren blir oppfordret til å skrive inn brukernavn og passord. Programmet er bygd opp etter prinsippet med PHP-koden og HTML-formularet på samme fil – jf. avsnitt B.3.5. For input-feltet for passordet er valgt `TYPE=PASSWORD` som forhindrer at passordet kommer fram i klartekst på skjermen. Når brukeren klikker på `SUBMIT`-knappen, forsøker programmet å logge seg inn i databasen. Hvis det ikke lykkes, blir skjermbildet stående slik at brukeren kan forsøke en gang til. Hvis innloggingen er vellykket, kommer det en melding om dette. Symbolet @ foran PHP-ORACLE-funksjonen `OCILOGON` sørger for at funksjonen er taus ("silent"), hvilket vil si at den ikke gir fra seg ordrike feilmeldinger når innloggingen mislykkes.



På en eller annen måte må vi ta vare på brukernavn og passord slik at de er tilgjengelige når vi skal logge inn i databasen fra andre PHP-programmer. For dette formålet oppretter vi en såkalt sesjon ("session") ved å kalle funksjonen `session_start()` og melde de to variablene `$bruker` og `$passord` inn i sesjonen med funksjonskallene `session_register('bruker')` og `session_register('passord')`.

Dette fører til at verdiene for disse to variablene blir oppbevart på et spesielt sted på tjenermaskinen, samtidig som det blir sendt en såkalt "cookie" med en peker til verdiene til klienten. Vi kan nå få tak i brukernavn og passord i andre PHP-programmer i samme sesjon ved å kalle funksjonen `session_start()` også i disse programmene – dermed blir de to variablene automatisk opprettet og initialisert med de riktige verdiene. Det er umulig å få tak i passordet på andre måter enn ved å kjøre PHP-programmer i samme sesjon. Sesjonen varer inntil man tar ned nettleseren.

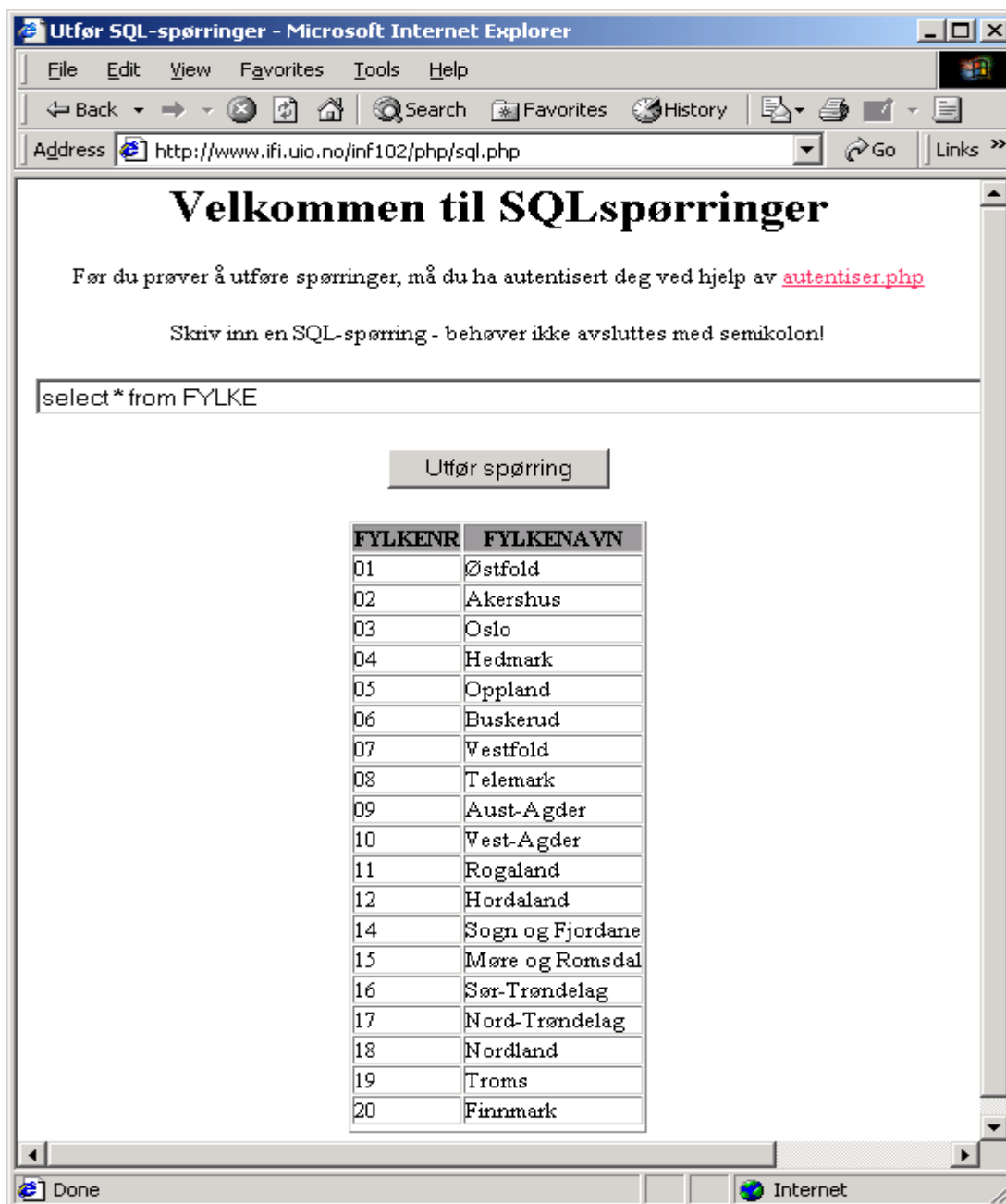
```
<?php // ligger på filen autentiser.php
session_start( );
session_register('bruker');
session_register('passord');

if(isset($_POST['bruker'])) $bruker = $_POST['bruker'];
else $bruker='';
if(isset($_POST['passord'])) $passord = $_POST['passord'];
else $passord='';
$msg = "Skriv inn brukernavn og passord og trykk Valider";
if((!$bruker)||(!$passord)) {
    $msg = "Skriv inn brukernavn og passord og trykk Valider!";
}else{
    if($conn = @OCILogon($bruker, $passord, "STUBAS")){
        $msg = "<strong>Brukernavn og passord OK!</strong>";
    } else {
        $msg = "<strong>Feil brukernavn og/eller passord. Prøv
igjen!</strong>";
    }
}
?>
<HTML>
<HEAD>
<TITLE>Autentisering</TITLE>
</HEAD>
<BODY>
<h1>Autentisering av databasebruker</h1>
<FORM ACTION="autentiser.php" METHOD="POST">
<table border=0>
<tr>
<td><strong>Brukernavn</strong></td>
<td><input type=text name=bruker size=10 maxlength=10></td>
</tr>
<tr>
<td><strong>Passord</strong></td>
<td><input type=password name=passord size=10 maxlength=10></td>
</tr>
<tr>
<td align=center><input type=submit value="Valider"></td>
<td align=center><INPUT TYPE="RESET" VALUE = "Tøm felt"></td>
</tr>
</table>
</form>
<p>
<?php print($msg); ?>
<hr>
<I>8. januar 2003. GS</I>
</BODY>
</HTML>
```

I mange lærebøker om PHP vil man finne eksempler på at autentisering blir gjort ved hjelp av HTTP "WWW-Authenticate header". Denne måten å gjøre det på egner seg best når man ønsker å begrense adgangen til hele programmet, ikke bare til databasen.

B.4.3 Et eksempelprogram – kjør SQL-spøringer

PHP-programmet nedenfor som ligger på filen `sql.php` setter opp et skjermbilde der brukeren kan skrive inn en SQL-spørring. Når brukeren klikker på knappen "Utfør spørring" blir spørringen sendt til databasen, og den resulterende tabellen vist på skjermen:



Programmet er bygd opp etter prinsippet med PHP-koden og HTML-formularet på samme fil – jf. avsnitt B.3.5, og er vist nedenfor. Vi starter opp med å inkludere de to filene `inc/db.inc` og `inc/gui.inc` for å få tilgang til hjelpefunksjonene. Deretter

henger vi oss på sesjonen med `session_start()` for å få tilgang til brukernavnet og passordet som brukeren har oppgitt under autentiseringen. Hvis vi har en ikke-tom spørring sendes den til databasen ved å kalle hjelpefunksjonen `baseQuery`, og resultatet formateres som en HTML-tabell ved hjelp av hjelpefunksjonen `visTabell`.

```
<?php // ligger på filen sql.php
include "inc/db.inc";
include "inc/gui.inc";
session_start( );

if(isset($_POST['query']))$query = $_POST['query'];else $query = '';
if($query) { // sjekk at vi har en ikke-tom query

$conn = baseLogon($bruker, $passord);

// Sender query til databasen, resultatet legges i variabelen $stmt
//

$stmt = baseQuery($conn, $query);

if(!$stmt) {
    print ("Feil i spørring".$msg);
} else {
    $html=visTabell($stmt, array( ));
    OCIFreeCursor($stmt);
}
baseLogoff($conn);
}
?>

<!-- Html-kode som vil lage tittel, overskrift, osv., >
<HTML>
  <HEAD>
    <TITLE>Utfør SQL-spørringer</TITLE>
  </HEAD>
  <BODY BGCOLOR="#FFFFFF" LINK="#2374fa" VLINK="#FF3366"
ALINK="#2374fa" TOPMARGIN="0">
    <CENTER>
      <H1>Velkommen til SQLspørringer</H1>
      Før du prøver å utføre spørringer, må du ha autentisert deg ved
      hjelp av
      <A href="autentiser.php">autentiser.php</A>
    <P>
    Skriv inn en SQL-spørring - behøver ikke avsluttes med semikolon!
    <P>

    <FORM METHOD="POST" ACTION = "sql.php">
    <INPUT NAME="query" SIZE="100" VALUE="<?php print($query); ?>">
    <P>
    <INPUT TYPE="SUBMIT" VALUE = "Utfør spørring">
    </FORM>

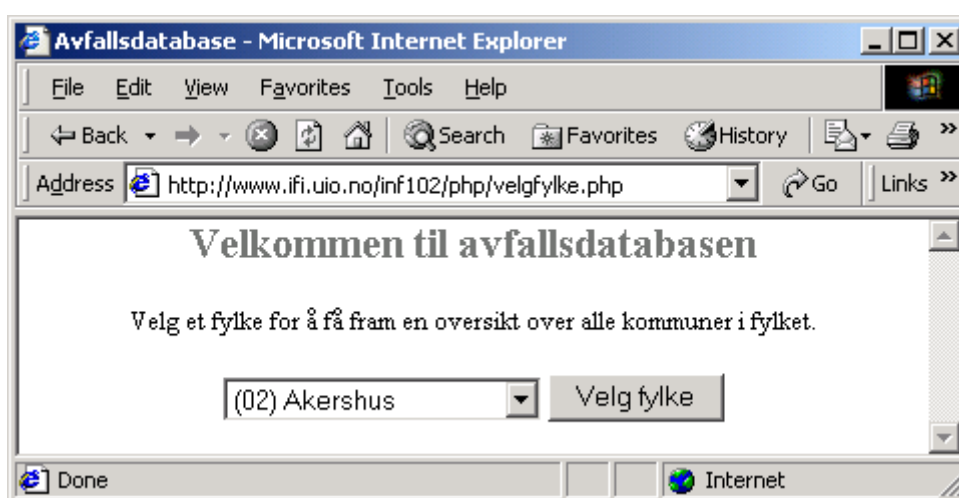
<?php
// skriv ut det pent formaterte resultatet av spørringen
print($html);
?>

  </CENTER>
</BODY>
</HTML>
```

B.5 Et litt større system

Som eksempel på et litt større system skal vi se på et sett med PHP-programmer som gir oss muligheter for å ta fram opplysninger fra Husholdningsavfallsdatabasen fra kapittel 4, figur 4-7, og oppdatere noen av disse opplysningene. PHP-programmene er skrevet slik at de illustrerer mange av mulighetene som finnes, og de kan brukes som utgangspunkt for å utvikle egne systemer.

Vi starter systemet ved å kalle programmet `velgfylke.php`. Før dette gjøres, må brukeren ha kjørt `autentiser.php` for å registrere brukernavn og passord. I et fullt ferdig system vil det være nærliggende å bake autentiseringen inn i et velkomstbilde med en link videre til `velgfylke.php`.



Brukeren velger et fylke på nedtrekksmenyen og klikker på knappen "Velg fylke". Dermed kalles programmet `kommuneoversikt.php` som gir en oversikt over kommunene i fylket, innbyggertall og avfallsmengder:

Oversikt over kommuner i Akershus fylke.

Klikk på pila for å kunne oppdatere opplysninger om den aktuelle kommunen.

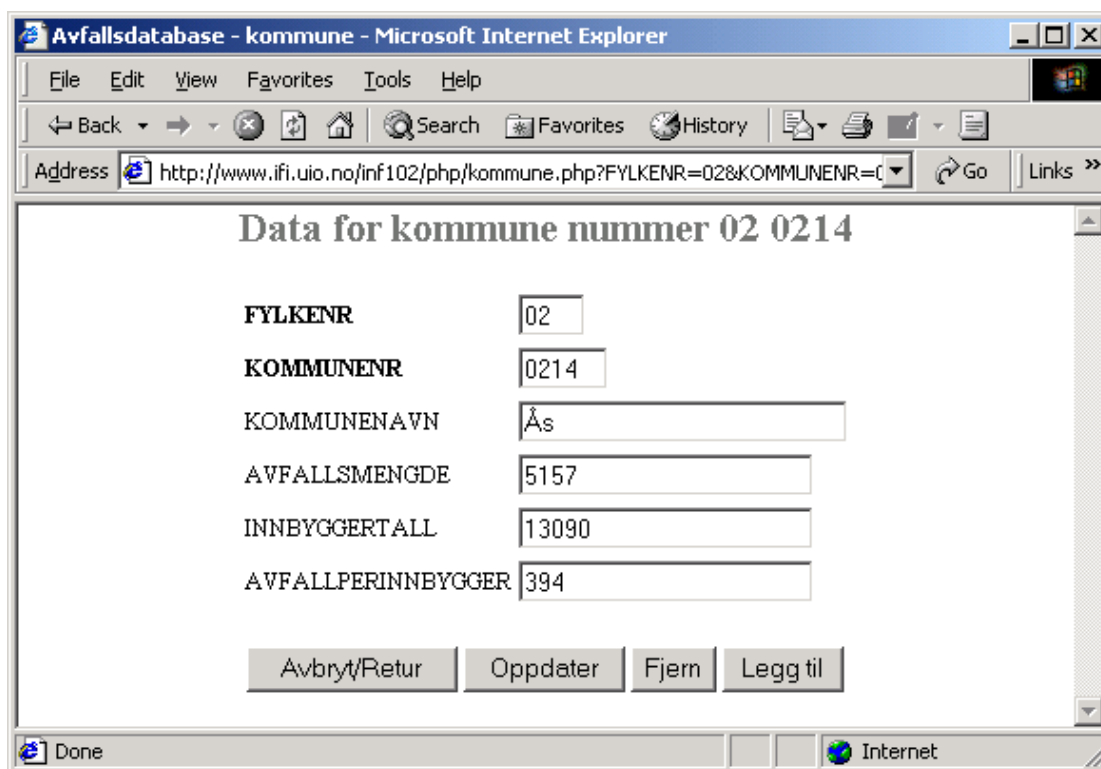
| FYLKENR | KOMMUNENR | KOMMUNENA VN | AVFALLSMENGDE | INNBYGGERTALL | |
|---------|-----------|----------------|---------------|---------------|---|
| 02 | 0211 | Vestby | 4869 | 11549 | ↓ |
| 02 | 0213 | Ski | 8226 | 24766 | ↓ |
| 02 | 0214 | Ås | 5157 | 13090 | ↓ |
| 02 | 0215 | Frogn | 4540 | 12517 | ↓ |
| 02 | 0216 | Nesodden | 4469 | 14691 | ↓ |
| 02 | 0217 | Oppegård | 7129 | 22611 | ↓ |
| 02 | 0219 | Bærum | 36024 | 99590 | ↓ |
| 02 | 0220 | Asker | 20009 | 47592 | ↓ |
| 02 | 0221 | Aurskog-Høland | 3412 | 12384 | ↓ |
| 02 | 0226 | Sørum | 4281 | 11879 | ↓ |
| 02 | 0227 | Fet | 3206 | 8895 | ↓ |
| 02 | 0228 | Rælingen | 5152 | 14296 | ↓ |
| 02 | 0229 | Enebakk | 3025 | 8393 | ↓ |
| 02 | 0230 | Lørenskog | 10417 | 28905 | ↓ |
| 02 | 0231 | Skedsmo | 13371 | 37102 | ↓ |
| 02 | 0233 | Nittedal | 6448 | 17892 | ↓ |
| 02 | 0234 | Gjerdrum | 1551 | 4304 | ↓ |
| 02 | 0235 | Ullensaker | 5458 | 19120 | ↓ |
| 02 | 0236 | Nes | 3982 | 15873 | ↓ |
| 02 | 0237 | Eidsvoll | 4847 | 16979 | ↓ |
| 02 | 0238 | Nannestad | 2412 | 8451 | ↓ |
| 02 | 0239 | Hurdal | 745 | 2611 | ↓ |

Tabellen inneholder 22 linjer.

[Legg til ny kommune](#)

Brukeren kan nå klikke på pilsymbolet til høyre på en av linjene i tabellen. Dette bringer oss videre til programmet `kommune.php` som viser fram et skjermbilde med opplysninger om den valgte kommunen. Attributtnavn for feltene som til sammen identifiserer kommunen (dvs. primærnøkkelen) er vis i fet skrift som et signal til brukeren om å være forsiktig med å endre dataene i disse feltene. Dersom brukeren klikker på linken "Legg til ny kommune" kommer vi til det samme skjermbildet, men med blanke felter.. Brukeren kan nå velge mellom å legge til nye kommuner, fjerne den viste kommunen, eller oppdatere innbyggertall, avfallsmengde eller

avfallperinnbygger for den viste kommunen. Oppdateringen av databasen utføres av `oppdaterkommune.php`. Deretter kommer vi tilbake til kommuneoversikten, der vi vil se resultatet av eventuelle oppdateringer.



The screenshot shows a Microsoft Internet Explorer window titled "Avfallsdatabase - kommune". The address bar contains the URL `http://www.ifi.uio.no/inf102/php/kommune.php?FYLKENR=02&KOMMUNENR=0214`. The main content area displays the title "Data for kommune nummer 02 0214" and a form with the following fields:

| | |
|--------------------|------------------------------------|
| FYLKENR | <input type="text" value="02"/> |
| KOMMUNENR | <input type="text" value="0214"/> |
| KOMMUNENAVN | <input type="text" value="Ås"/> |
| AVFALLSMENGDE | <input type="text" value="5157"/> |
| INNBYGGERTALL | <input type="text" value="13090"/> |
| AVFALLPERINNBYGGER | <input type="text" value="394"/> |

Below the form are four buttons: "Avbryt/Retur", "Oppdater", "Fjern", and "Legg til". The status bar at the bottom shows "Done" and "Internet".

PHP-koden for dette systemet er å finne på filene `velgfylke.php`, `kommuneoversikt.php`, `kommune.php` og `oppdaterkommune.php`.

Programmene er skrevet i henhold til en trelagsarkitektur slik at det skilles nokså klart mellom databaseoperasjoner, applikasjonslogikk og brukergrensesnittoperasjoner. Data bringes fra databasen til brukergrensesnittoperasjonene gjennom bufferen `$stmt`.

For at programmene ikke skal bli for store og uoversiktlige, er de ikke bygd ut med tanke på å fange opp alle mulige feilsituasjoner. Dersom brukeren bruker systemet på en slik måte at det forårsaker feilaktige spørringer og oppdateringer mot databasen, for eksempel ved å legge inn kommuner som allerede fins, vil dette føre til mer eller mindre forståelige feilmeldinger direkte fra databasehåndteringssystemet.

B.6 Sikkerhet for databaser under PHP

Databaser som er direkte eller indirekte tilgjengelig over Internett kan være utsatt for angrep: Uvedkommende kan forsøke å lese data de ikke burde ha tilgang til, og noen kan til og med forsøke å endre innholdet i databasen. Her skal vi se på hvordan vi med relativt enkle midler kan beskytte databasen, selv om sikkerheten mot innbrudd ikke blir 100 %.

B.6.1 Tilgang til databasen

Tilgangen til databasen oppnås gjennom å oppgi brukernavn og passord. En bruker har absolutt alle rettigheter til sitt eget område. Et PHP-program som kan startes av hvem som helst over Internett bør kun ha begrensede rettigheter – helst bare de som trengs for at programmet skal kunne kjøres. Det ideelle er derfor å opprette et antall brukernavn med tilhørende passord som brukes bare av PHP-programmene.³, og som av hovedbrukeren gis de nødvendige rettigheter til hovedbrukerens tabeller. Disse "PHP-brukerne" vil trolig ikke ha noen tabeller på eget område. Husk at hovedbrukeren ved hjelp av virtuelle tabeller kan bestemme svært nøyaktig hva en annen bruker skal kunne se av data i databasen.

B.6.2 Skjuling av brukernavn og passord

Vi kan lage systemet slik at det ber om brukernavn og passord til databasen. Dette er ikke spesielt brukervennlig. Hvis vi attpåtil oppretter en rekke "PHP-brukere" med ulike rettigheter som skissert i forrige avsnitt, vil det være sikkerhetsmessig forkastelig å offentliggjøre brukernavn og passord for alle disse.

Alternativet er å la PHP-programmet få tak i brukernavn og passord fra et hemmelig sted. Dette kan gjøres ved å legge inn brukernavn og passord som konstanter på en include-fil:

```
<?
define ("BRUKER", "brukernavn");
define ("PASSWORD", "passord");
?>
```

Include-filen legges i en mappe i et område av filsystemet som er utilgjengelige for andre brukere av filsystemet⁴. Systemet er satt opp med en ikke-publisert standard sti fram til denne mappen. Include-filen må kunne leses (den må altså ha UNIX-tilgangskode rw-r--r-- eller 644), mens mappen skal kun ha eksekveringstilgang for å hindre noen i å se hvilke filer som ligger i mappen (mappen må altså ha UNIX-tilgangskode rwx--x--x eller 711). PHP-programmet kan nå få tak i konstantene med `include "brukernavnogpassordfil"`.

B.6.3 Sikring av SQL-kommandoer

En SQL-kommando som sendes ut fra et PHP-program er typisk satt sammen av en fast del som ligger i programmet, og en parametrisert del som brukeren legger inn i et skjermbilde. Dette åpner for at en bruker med skumle hensikter kan få generert SQL-kommandoer som gjør noe helt annet enn det som var hensikten. Tegnsekvensen -- som

³ I forbindelse med INF102 er dette driftsadministrativt vanskelig

⁴ I Ifi's filoppsett er slike områder ikke tilgjengelig

bevirker at resten av SQL-kommandoen betraktes som kommentar, er spesielt anvendelig ved slike innbruddsforsøk.

Mottiltaket her er å la PHP-programmet sjekke grundig at de verdiene som brukeren legger inn i skjermbildet er slik som de forventes å være, for eksempel at tabellnavn ser ut som tabellnavn, at numeriske verdier virkelig er numeriske osv. Spesielt bør programmet sjekke at det ikke finnes umotiverte -- i verdiene. Se også <http://www.php.net/manual/en/security.database.php> om SQL Injection.

