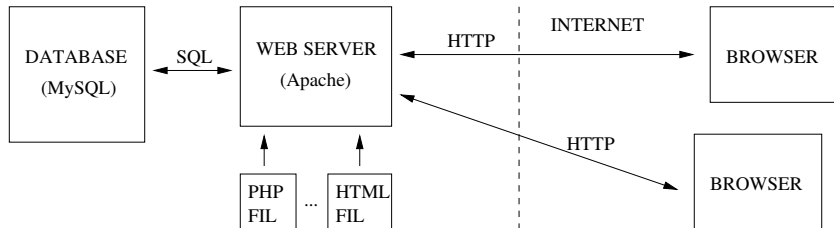


- ▶ Hvad har vi lært indtil nu?
- ▶ Tilgang til MySQL databaser fra PHP-scripts
- ▶ Konstruktion af databasestøttede web-sites
- ▶ Eksempel: Mailing-list
- ▶ `auto_increment` i MySQL

Hvad har vi lært indtil nu?

Oversigt:



En PHP-fil:

```
<body>
  <?php
    echo "<b>Hello</b> ";
    echo "<i>WORLD</i>";
  ?>
</body>
```

Indtil nu:

- ▶ Variabler, tal, strenge og arrays
- ▶ Beregninger
- ▶ `if`-sætninger og løkker
- ▶ Funktioner og kodegenbrug
- ▶ Indhentning af data fra brugere med forms
- ▶ Check af brugerindtastninger med regulære udtryk
- ▶ Grundlæggende SQL

Etablering af forbindelse:

maskinen *mysql.itu.dk* for bruger *holbech* med password *pass* etableres med følgende kald af PHP-funktionen `mysql_connect()`:

```
$conn = mysql_connect("mysql.itu.dk", "holbech", "pass");
```

Valg af database:

Databasen med navn *DSDStest* vælges ved følgende kald af funktionen `mysql_select_db()`:

```
$db = mysql_select_db("DSDStest", $conn);
```

Begge funktioner returnerer en `boolean` (sand/falsk værdi) der angiver om alt gik glat.

Udførelse af SQL queries:

Datamanipulationskommandoer udføres ved brug af funktionen `mysql_query`.

Ved udførelse af SQL queries, med PHP-funktionen `mysql_query`, returneres en værdi der repræsenterer rækkerne i resultatet.

Derefter kan man ved brug af funktionen `mysql_fetch_array` håndtere rækkerne en efter en, som numerisk og/eller associative arrays.

Filen: mysql_fetch_array.php

```
$db = mysql_connect("mysql.itu.dk", "holbech", "pass");
mysql_select_db("DSDStest", $db);

$result=
    mysql_query("SELECT name,zip FROM mytable WHERE zip=2200");
//Vi looper igennem alle valgte rækker
while($row = mysql_fetch_array($result) ){
    echo "<p>$row[0] - $row[1]</p>";
}

$result=
    mysql_query("SELECT name,zip FROM mytable WHERE zip=2200");
//En gang til, som associative array
while($row = mysql_fetch_array($result) ){
    echo '<p>'. $row['name']. ' - '. $row['zip']. '</p>';
}
```

Én fremgangsmåde:

1. Konstruktion af datamodel

- ▶ Hvilken information skal gemmes og hvordan skal den repræsenteres?
- ▶ *Dette er den svære del!*

2. Udvikling af data-transaktioner, SQL

- ▶ Hvordan indsættes data i databasen?
- ▶ Hvordan udtrækkes data fra databasen?

3. Konstruktion af web-forms og site-map

- ▶ Brugergrensefladen er HTML-kode (forms)

4. Konstruktion af PHP-filer til implementation af data-transaktioner

- ▶ *Dette er den nemme del!*

Bemærk: Jo mere tid der bruges på de første trin, jo lettere bliver de sidste!

Kravspecifikation: Med mailing-list systemet skal vi kunne oprette en liste af navne og emails til brug for udsendelse af emails. Den samme fælles liste vedligeholdes af alle der anvender systemet. Den eneste information som gemmes er navne og emails.

Trin 1: Datamodellen

```
CREATE TABLE IF NOT EXISTS maillist (  
    email      VARCHAR(100) NOT NULL,  
    name       VARCHAR(100) NOT NULL  
);  
CREATE UNIQUE INDEX email ON maillist (email);
```

Vi angiver, at der ikke er to personer der kan anvende den samme email, dvs. email er `unique`

Vi påtvinger navneoplysninger og oplysninger om email; `NOT NULL`

Trin 2: Data-transaktioner

Af systemet kræves følgende former for data-transaktioner:

- ▶ Indsættelse af nye emails og navne:

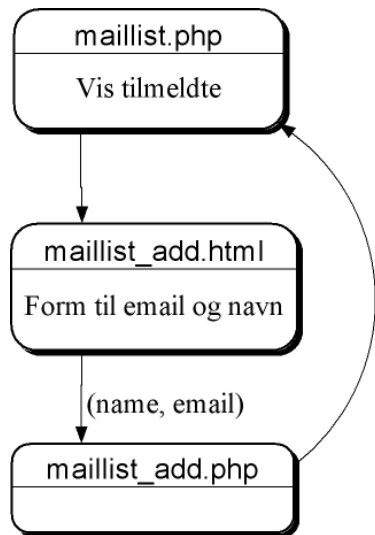
```
insert into maillist (email,name) values  
('nh@itu.dk', 'Niels Hallenberg');
```

```
insert into maillist (email,name) values  
('holbech@itu.dk', 'Jonas Holbech');
```

```
insert into maillist (email,name) values  
('gates@microsoft.com', 'Bill Gates');
```

Vi skal senere udvide kravene til systemet!

Trin 3: Konstruktion af web-forms og site-map



- ▶ Kasserne i diagrammet repræsenterer scripts.
- ▶ (Parateserne) de værdier der bliver sendt til scriptet
- ▶ For hver kasse er det angivet hvad filen hedder samt hvad der eventuelt vises til brugeren

Opbygning af web-forms: maillist_add.html

```
<h2>Add Yourself to the Mailing List</h2>
<form action="maillist_add.php" method="post">
  <table>
    <tr><td>Email:</td><td><input name="email"></td></tr>
    <tr><td>Name:</td><td><input name="name"></td></tr>
    <tr><td colspan="2">
      <input type="submit" value="Add">
    </td></tr>
  </table>
</form>
```

Bemærk:

- ▶ Filen `maillist_add.php` er *action* for formen
- ▶ formen indeholder to felter med navne `email` og `name`

Trin 4: Konstruktion af PHP-filer

Filen maillist.php — visning af email-adresser:

```
<?php
// Etabler databaseforbindelse
$db = mysql_connect("mysql.itu.dk", "testing_dsds", "***");
mysql_select_db("DSDStest", $db);

// Udtræk rækker fra tabel
$rows = mysql_query("select email, name from maillist");
echo '<ul>';

// Kør igennem alle rækkerne i tabellen, en efter en
while ( $row = mysql_fetch_array($rows) ) {
    echo "<li> <a href='mailto:$row[0] '$row[1]</a></li>";
}
?>
<li><a href="maillist_add.html">Add Yourself</a></li>
</ul>
```

Eksempel: mailing-list — fortsat

Filen maillist_add.php — tilføjelse af email-adresse:

```
<?php
// "hent" formvariabler
$name = $_REQUEST['name'];
$email = $_REQUEST['email'];

// Etabler databaseforbindelse
$db = mysql_connect("mysql.itu.dk", "testing_dsds", "***");
mysql_select_db("DSDStest", $db);

// Indsæt data i database
mysql_query("insert into maillist (email, name)
            values ('$email', '$name')");

// Hop tilbage til maillist.php
header("Location: maillist.php");
?>
```

- ▶ Ved at kalde funktionen `header` med `Location: maillist.php` sendes der information til browseren (med HTTP) om at browseren skal efterspørge filen `maillist.php` på serveren.
- ▶ Da dette sker hurtigt — og uden brugerinteraktion — er resultatet at den opdaterede mailing-liste bliver vist til brugeren.
- ▶ Er der nogle uhensigtsmæssigheder og mangler i forgående scripts?

Brug af include-fil til etablering af forbindelse

Ved at benytte include-funktionen undgår vi at password-information står i alle filer: mydb.php

```
function error ( $msg ) {
    echo "<h2>Error in PHP script</h2><p>$msg</p>";
    exit();
}

// funktion til etablering af forbindelse til database
function mydb_connect() {
    $db = mysql_connect("mysql.itu.dk", "holbech", "****");
    if ( $db == 0 ) {
        error ("Ingen forbindelse til databasen");
    }
    $database = "DSDStest";
    if ( mysql_select_db($database, $db) == 0 ) {
        error ("Kunne ikke vælge databasen: '$database'");
    } }
}
```

Bemærk: Vi tester returværdierne fra `mysql_connect` og `mysql_select_db`

Lad os udvide vores mailing-list eksempel således at det bliver muligt at slette navne fra listen

Vi kigger på de fire punkter igen:

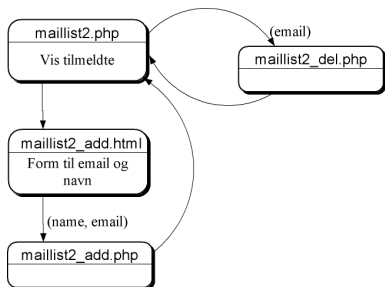
Trin 1: Datamodellen er uændret (tabel `maillist`)

Trin 2: Følgende data-transaktion tilføjes:

- ▶ Sletning af email og tilknyttet navn

```
delete from maillist
where email = 'gates@microsoft.com';
```

Trin 3: Konstruktion af site-map:



- ▶ Filen `maillist2_del.php` sletter en række i tabellen
- ▶ Filen forventer en formvariabel `email`, som overføres i et link i filen `maillist2.php`:
`ala`
`maillist2_del.php?email=g@m.dk`

Trin 4: Konstruktion af PHP-filer

Filen maillist2.php:

```
<?php
include("mydb2.php"); // Inkluder utilities
mydb_connect();      // Connect til database
...
// Udtræk rækker fra tabel
$rows = mysql_query("select email, name from maillist");
// Kør igennem alle rækkerne i tabellen, en efter en
while ( $row = mysql_fetch_row($rows) ) {
    // Udskriv enkelt række
    echo "<li><a href='mailto:$row[0] '>$row[1]</a>
        <a href='maillist2_del.php?email=$row[0] '>delete</a></li>";
}
?>
...
</ul>...
```

Udvidelse af eksempel mailing-list — fortsat

Filen maillist2_del.php:

```
<?php
include("mydb2.php");          // Inkluder utilities

// Check formvariabler
if ( ereg("[a-zA-Z][a-zA-Z.0-9-]*@[a-zA-Z.0-9-]+", $email)==0){
    error("Go back and enter an email!");
}
mydb_connect();              // Connect til database

// Slet række
mysql_query("delete from maillist where email = '$email'");
// Hop til hovedsiden
header("Location: maillist2.php");
?>
```

Bemærk: Vi checker at formvariablen `email` er en valid email-adresse

Næste gang: Vi ser bl.a. på hvordan emails sendes til de tilmeldte

Eksempel:

I MySQL kan man benytte `auto_increment` til at generere nye identitetstal automatisk når der indsættes nye rækker i en tabel

```
CREATE TABLE Users (id INT AUTO_INCREMENT PRIMARY KEY,  
                    name VARCHAR(100) NOT NULL);  
  
INSERT INTO Users (name) VALUES ('Martin Elsman');  
INSERT INTO Users (id, name) VALUES ('', 'Jonas Holbech');
```

Herefter har Martin fået id 1 og Jonas id 2

Andre databasesystemer tilbyder tilsvarende funktionalitet

Hvorfor er autogeneratede id'er en god ide?

Konstruktion af kommentarservice

Giv mulighed for at læsere af dine web-sider kan kommentere siderne...