A8: a Web Service for Organizing Parties

[DSDS E2012 - Hand-in deadline: Friday, November 16, 2012 at 08:59]

Goal

In this assignment you will construct three PHP script files that 1) use a pair of MySQL tables to maintain a database of available parties, and 2) give the user the possibility to sign up for specific parties through an HTML form.
The tables to be created are `parties` and `guests`, designed to contain data about, yes, parties and guests. The php scripts to be designed are:

- `list_parties.php` (shows info about all upcoming parties to the user, including the number of people that have signed up for each);
- `list_attending.php` (shows detailed info about a specific party and provides a form which the user can use to sign up her/himself); *and*
- `save_attending.php` (validates the user input and inserts the data into a new record in the guest SQL table).

Preparation: copy files which you will `include` later

To speed up the process of solving this assignment, we recommend you to make use of some functions defined in PHP files located here:

- [http://www.itu.dk/people/brabrand/DSDS/includes/]

As you might recall, we cannot inspect the content of PHP files by simply opening them in a browser (because PHP scripts get executed on the server). That is, you cannot see what such files contain by simply clicking on them in the browser. You can however use the `show.php` script (available in the same place) to see the PHP source code of the files, passing the filename of the file you want to inspect as an argument in the URL. If you for instance want to see the content of the `fn_mydb_connect.php` file, paste this into your browser:

- [http://www.itu.dk/people/brabrand/DSDS/includes/show.php?file=fn_mydb_connect.php]

You could in theory include the files (using `include` statements in your PHP scripts) directly from where they are right now but since you will probably have to edit at least one of them (`fn_mydb_connect.php`), it is better if you copy the files to your own local directory so that you can modify them whenever you need to. Create an `includes/` directory in the root of your local student directory, i.e. `e2012/DSDS/`*`username`*`/includes/` and copy the files to that location using whatever method you like. One way is to use the `show.php` method mentioned above to display the PHP contents of each of the files (`fn_headerfooter.php`, `fn_input_validation.php`, and `fn_mydb_connect.php`) in your browser and then copy and paste the code into your text editor, giving the resulting files the same name as the originals.

Once you have the files (fn_headerfooter.php, fn_input_validation.php, and fn_mydb_connect.php) in your includes/ directory, you are all set to start working on this assignment.

Exercise 8.1:Create the `parties` and `guests` SQL tables and insert some data

- Destination file: A8/create_tables.sql

The database part of the party web site consists of two MySQL tables, each of them consisting of four fields (columns), as illustrated in a so-called "ER diagram" below:

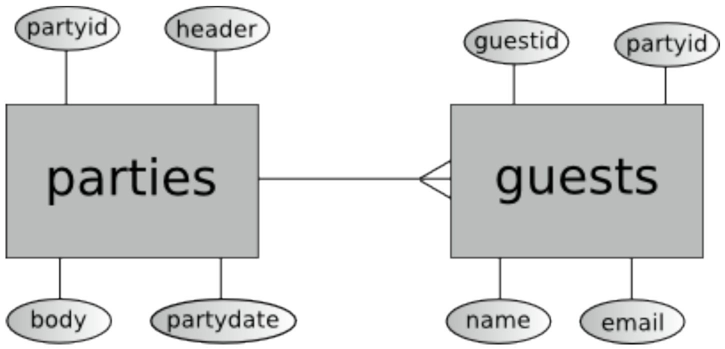


Figure 1: ER –Diagram of "parties" and "guests" (illustrating fields and relationship).

Here is a more detailed description of the tables and the datatypes of their fields:

parties:

- partyid (integer, primary key, auto_increment, not null), a unique id
- header (varchar, not null) party name/short description
- body (text), longer description of the party
- partydate (date), the date for the party (so we can show upcoming parties, avoid showing those who have already passed, sort them, etc.)

guests:

- guestid (integer, primary key, auto_increment, not null), a unique id
- partyid (integer, not null) foreign key pointing to parties.partyid
- name (varchar, not null) the name of the party participant
- email (varchar, not null) the email of the party participant

Create the two tables according to the specification above using the MySQL text client and CREATE TABLE commands. (You will need to connect to your personal database on the mysql.itu.dk server first. If you don't remember how, take a look in assignment 6 or 7 where you can find step-by-step instructions how to do this.)

Insert 4 parties and 2-3 guests into the database using the INSERT INTO command. Make sure that all of the parties are in the future.

Check that your tables look fine by performing the following queries:

- `SELECT * FROM parties;`
- `SELECT * FROM guests;`

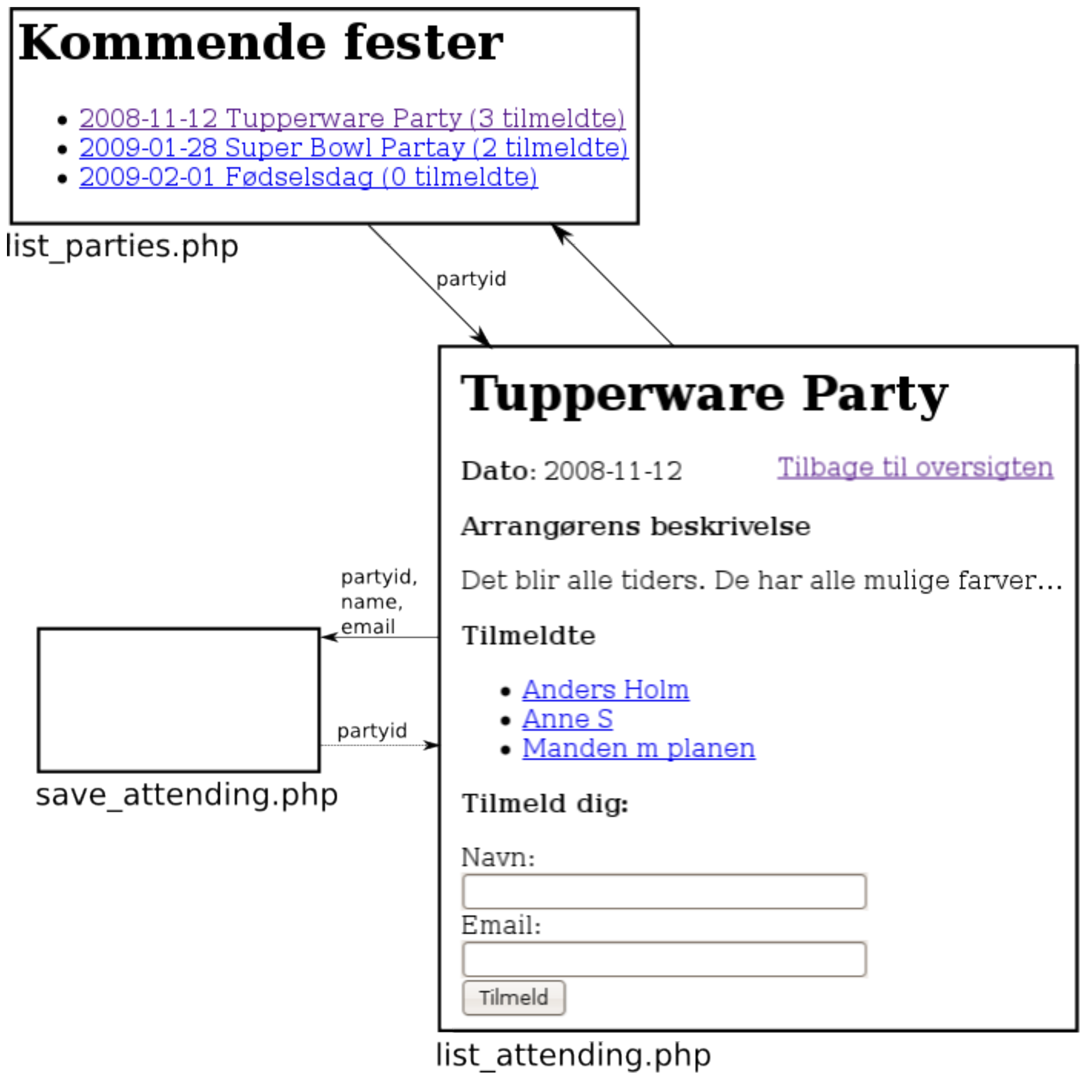
Save your SQL commands (CREATE TABLE, INSERT INTO, and SELECT * FROM) in the file:

- `A8/create_tables.sql.`

Exercise 8.2: list_parties.php

- Destination file: `A8/list_parties.php`

In this exercise you will create the first out of a total of three PHP scripts. The three PHP scripts relate to each other as illustrated in the sitemap/flowchart below:



Kommende fester

- 2008-11-12 Tupperware Party (3 tilmeldte)
- 2009-01-28 Super Bowl Partay (2 tilmeldte)
- 2009-02-01 Fødselsdag (0 tilmeldte)

Tupperware Party

Dato: 2008-11-12 Tilbage til oversigten

Arrangørens beskrivelse

Det blir alle tiders. De har alle mulige farver…

Tilmeldte

- Anders Holm
- Anne S
- Manden m planen

Tilmeld dig:

Navn:

Email:

The specific purpose of the `list_parties.php` script is to show a list of all upcoming parties to the user, including the number of people that have signed up for each (see the diagram above for an example of how the resulting web page could look). For each of the listed parties, there should be a clickable link that links to the `list_attending.php` script. (That script, which you will create in the next exercise, will give the user more detailed info about the specific party including the names of all current participants as well as providing a form where the user can enroll her/himself.)

Requirements for `list_parties.php`

The script should show:

- All upcoming parties that are stored in the parties table. [Hint: use a SELECT query WHERE partydate >= NOW().] Remember to add a suitable ORDER BY construct to your SQL query so that the list starts with the parties closer in time and show those occuring later at the end of the list.
- The total number of enrolled persons for each party. [Hint: use COUNT and LEFT JOIN.]
- The short description of the party (header).

The short description of the party (the `header` field) should be a link to the `list_attending.php` script where the partyid of the specific party is transferred directly in the URL, something like `<a href='list_attending.php?partyid=2'>Party name (2 attending)</a>`

You will need to use the mysql_query() and mysql_fetch_array() functions discussed at the lecture today, resulting in code looking something like $result=mysql_query("SELECT...") followed by $row=mysql_fetch_array($result).

Connect to your database using the `mydb_connect()` function defined in `includes/fn_mydb_connect.php`

However, before the PHP server can execute `mysql_query()` and `mysql_fetch_array()` statements, your script needs to have established a connection to your database using `mysql_connect()` and `mysql_select_db()`. Instead of letting all the scripts use these commands for establishing the database connection, and all of these scripts reveal the password of your database, you can use the `mydb_connect()` function to take care of this. All you need to do in order to be able to use this function is to make sure to include the file that defines it: `fn_mydb_connect.php`. If you have done the "preparative action" described in the beginning of this document, you should have this file in your `includes/ directory`.

Now, if you open that file, you will see that the name of the username, the password, and the name of the database all have dummy values. You need to change those values so that the `mydb_connect()` function connects to ***your*** database. Done? Then, use the following short and simple line in your `list_parties.php` script to connect to your database: `mydb_connect();`

The neat thing is that now a) your sensitive data (in particular the database password) is stored in only one file instead of several, b) the code in each of your scripts becomes shorter and easier to read.

Among the three files you copied to your `includes/` directory, you also have the fn_headerfooter.php file which contain simple but useful functions for generating an XHTML 1.0 Strict header (`show_header()`) and closing the HTML part of the file (`show_footer()`)

If you have followed the instructions so far, your list_parties.php will probably look something like this:

```
<?php
    include("../includes/fn_headerfooter.php");
    include("../includes/fn_mydb_connect.php");
    //...
    show_header("let's party");
    //...
    mydb_connect();
    //...
    $result = mysql_query("SELECT ... FROM ...");
    while ... mysql_fetch_array($result) ... {
        echo "<li> <a href=...  ";
    }
    //...
    show_footer();
?>
```

Once you think you have everything you need in your `list_parties.php`, try it out by navigating your browser to the page generated by the PHP script (`A8/list_parties.php`) and start the debugging. When everything works, you should see something similar to the `list_parties.php` box pictured in the sitemap earlier in this document, only that it is going to be the data inserted by you in exercise 8.1 that will be listed. Congratulations – you have created your first PHP script that fetches data from a MySQL database!

Don't forget to check that the HTML code generated by your script is reasonable.

Exercise 8.3: list_attending.php

- Destination file: `A8/list_attending.php`

In this exercise you will create the `list_attending.php` script. The script receives a `partyid` from `list_parties.php`. This `partyid` signifies what party that should be shown in detail. Based on the `partyid`, we retrieve the relevant fields from the `parties` table and print them out in a way similar to what is shown in the sitemap. After that, all the names of the guest that have enrolled to the party is printed out. Finally, a form is presented, allowing the user to add her/himself (or a friend) to the party. All in all, this script, just like the previous one and the next one in this assignment, have the same basic structure of PHP scripts accessing MySQL databases illustrated in the appendix of this document.

Start with storing the value of `$_REQUEST['partyid']` into a variable.
Using a SELECT FROM query, retrieve the record corresponding to the `partyid` from the `parties` table and print out the relevant information (`header`, `body`, `partydate`).
Retrieve all guests enrolled in the party with the specific `partyid` and print them out, one at a time (you will need a `while` loop for this). For each of the guests, the guest name should be a "mailto:" link of the kind `<a href='mailto:email'>name</a>`. so that if the user clicks on the name, the user's email client will invoke an empty email addressed

Finally, there should be a form:

- The `action` of the form should be `save_attending.php`
- `method` should be `post`
- The form contains 4 fields:
    - name (input type='text'), the name of the guest
    - email (input type='text'), the email of the guest
    - partyid (input type='hidden'), the id of the party, which we also need to send, or the receiving script will not know to what party the person wants to enroll in
    - and the submit button

Save your solution in `A8/list_attending.php`.

Exercise 8.4: save_attending.php

- Destination file: `A8/save_attending.php`

In this exercise you will create the last of the three PHP scripts you need to make this party web service complete: `save_attending.php`.
The script receives partyid, name, and email from `list_attending.php`. The script should use regular expressions to validate the values of these variables (make sure partyid is a number, make sure the name is a name, and that email is indeed an email address). This is to ensure that the data makes sense and also to prevent the database from being hacked. If you

do not want to construct the regular expressions yourself, you can make use of the functions defined in `fn_input_validation.php` in order to do the job.

Start by storing the content of the `$_REQUEST[]` array received from the `list_attending.php` script into suitable variables, i.e. something like `$name = $_REQUEST['name'];`

After that, your script should validate the content of the variables. If the data in any of the variables do not conform to what is expected, e.g. if the name is not a string with characters [A-Za-z] with space(s) in between, or if a variable is plain empty, we pull the handbrake, show an error message and stop the script (for instance using the function `error()` defined in the `fn_input_validation.php` file.

If everything looks fine, a new record is inserted in the table guests and we send the user back to the `list_attending.php` including the current partyid in the URL. The directing of the user's web browser to the specific `list_attending.php` URL is done using the `header()` command as shown on the lecture earlier today, i.e. something like this:

```
...
header("Location: list_attending.php?partyid=$partyid");
exit();
```

Now, eventually, after some debugging, you should have a fully functioning web site that allow people to sign up for parties! :-)   Save your solution in `A8/save_attending.php`.

(We skip adding the functionality of adding parties themselves or allowing people to sign off from parties they previously signed up for. If you have managed to get this far with this assignment, you are in fact fully capable to extend the web site with these functionalities if you would like to in the future. You have all the knowledge needed to do this.)

Checklist

If you have done all the exercises of this assignment, you should have the following four files in your `A8/` directory:

- `create_tables.sql`
- `list_attending.php`
- `list_parties.php`
- `save_attending.php`

And the following three files in your `includes/` directory:

- `includes/fn_headerfooter.php`
- `includes/fn_input_validation.php`
- `includes/fn_mydb_connect.php`

Remember to validate all your PHP files, making sure they generate valid HTML code.

Appendix: A general template for PHP scripts accessing a MySQL database

Both in this assignment and in future ones, you will construct PHP scripts that in one way or the other accesses your MySQL database. These scripts will all more or less have the following basic structure:

```
-------------------------------------
// - Preliminary html

/* --- If there are any form variables to be collected:
- Collect form variables
- Validate form variables
*/

/* If form variables are ok (or none collected):
- Connect to database
*/

/* Do something with database (SELECT, INSERT, DELETE, UPDATE etc.)
SELECT: Use while loop + link to another file OR
INSERT/ DELETE / UPDATE: Just one row + header("Location: "  .. ");
*/

// etc.

// - Closing html
---------------------------------------
```