Databases

PHP Data Objects (PDO)

Session variables

Files

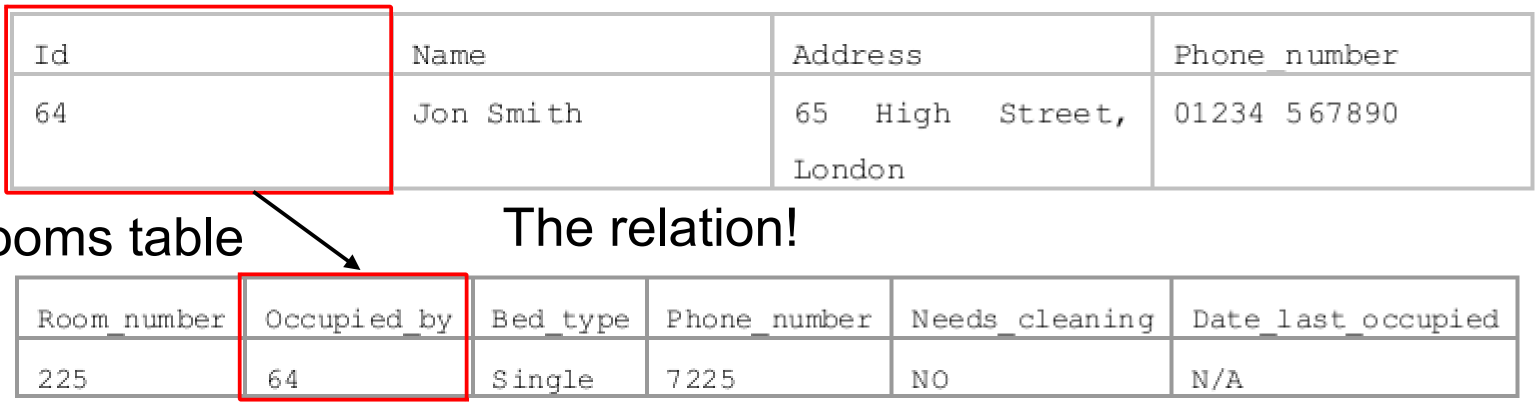
Databases for web

- In order to do server-based programs we need the ability to interrogate or ”talk” with databases
  - Usually the language SQL (Structured Query Language) is used
- There is a lot of RDBMS (Relational DataBase Management Systems) available – from free to expensive
  - Client-server
    - PostgreSQL, MySQL
    - Microsoft SQL server, Oracle 11g, IBM DB2, Sybase SQL Anywhere
  - Linked into the application
    - SQLite, Microsoft Access (Jet)
- Usually the web administrator connect via the web browser and manage the database via some server-side framework.
  - Other methods is by console or a special management software
- Many hosting services are database enabled since there is little use of server-side programming otherwise!

Databases, Tables, Fields, Rows and Columns

- A **table** is a collection of **rows** and **columns** just like a spreadsheet in for example MS Excel
- A **field** is an individual cell in a **table**
- A **database** is a collection of **tables**
- Normalization
  - The golden rule of database design is: "don’t store anything more than once. Instead, store it just once, give it a unique reference number, and refer to it by that number in the future"
- Example of a hotel management system

Customers table



Id	Name	Address	Phone_number
64	Jon Smith	65 High Street, London	01234 567890

Rooms table

The relation!

Room_number	Occupied_by	Bed_type	Phone_number	Needs_cleaning	Date_last_occupied
225	64	Single	7225	NO	N/A

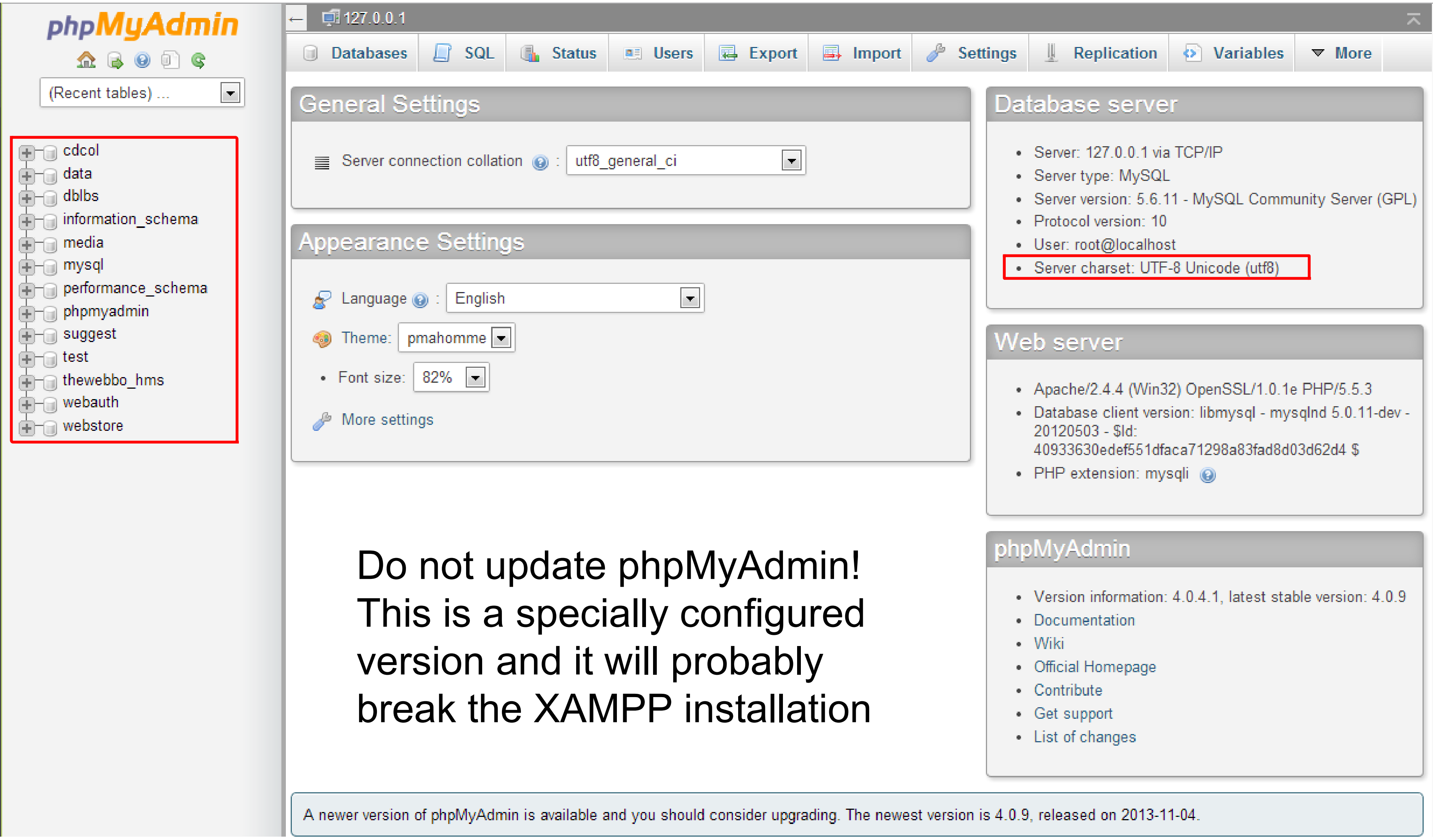
Referential Integrity

- A normalised database
  - Each type of data item is held in a separate table with a unique id - a very efficient way to store and manage information!
- It is important to ensure that you don't end up with tables that refer to non-existing information
  - In database terminology, this means making sure that you maintain the **referential integrity**
- Example: consider if we delete the customer Jon Smith?
- Different solutions
  - Delete the customer from the table. In addition, also delete every affected row in every other table (with triggers or manually)
  - Replace the contents of customer's record with text such as "deleted"
  - Add a field to the customers table called, is_live, which specifies whether this is a live customer record or not

Creating a hotel management database with phpMyAdmin

- LAMP/WAMP (Linux/Windows) server
- Apache, MySQL and PHP (some helper programs)
  - http://en.wikipedia.org/wiki/LAMP_(software_bundle)
  - http://en.wikipedia.org/wiki/Comparison_of_WAMPs
- Install the portable XAMPP-USB-LITE version
  - Download, unpack, read readme_en.txt and follow 4 simple instructions, finished! (if you are lucky!)
- Configure the database with phpMyAdmin
- On the left hand side are the databases
- The existing databases are part of the inner workings of MySQL or phpMyAdmin exept the test database
- Create a new database wit the name thewebbo_hms and Collation utf8_***
  - cs = case sensitive, ci = case insensitive

phpMyAdmin



- cdcol
- data
- dblbs
- information_schema
- media
- mysql
- performance_schema
- phpmyadmin
- suggest
- test
- thewebbo_hms
- webauth
- webstore

General Settings

Server connection collation: utf8_general_ci

Appearance Settings

Language: English

Theme: pmahomme

- Font size: 82%

More settings

Database server

- Server: 127.0.0.1 via TCP/IP
- Server type: MySQL
- Server version: 5.6.11 - MySQL Community Server (GPL)
- Protocol version: 10
- User: root@localhost
- Server charset: UTF-8 Unicode (utf8)

Web server

- Apache/2.4.4 (Win32) OpenSSL/1.0.1e PHP/5.5.3
- Database client version: libmysql - mysqlnd 5.0.11-dev - 20120503 - $Id: 40933630edef551dfaca71298a83fad8d03d62d4 $
- PHP extension: mysqli

phpMyAdmin

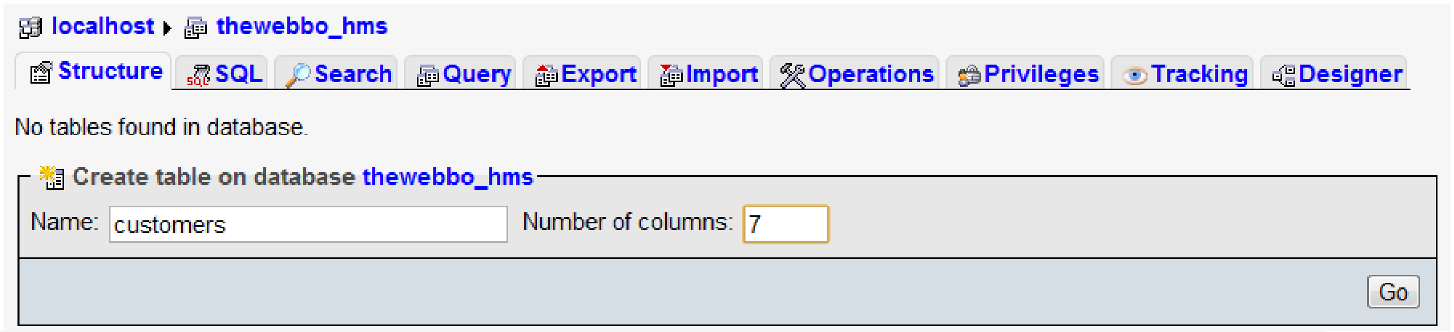
- Version information: 4.0.4.1, latest stable version: 4.0.9
- Documentation
- Wiki
- Official Homepage
- Contribute
- Get support
- List of changes

Do not update phpMyAdmin! This is a specially configured version and it will probably break the XAMPP installation

A newer version of phpMyAdmin is available and you should consider upgrading. The newest version is 4.0.9, released on 2013-11-04.

Hotel management DB 1

- Create a table with the UI



localhost ▸ thewebbo_hms

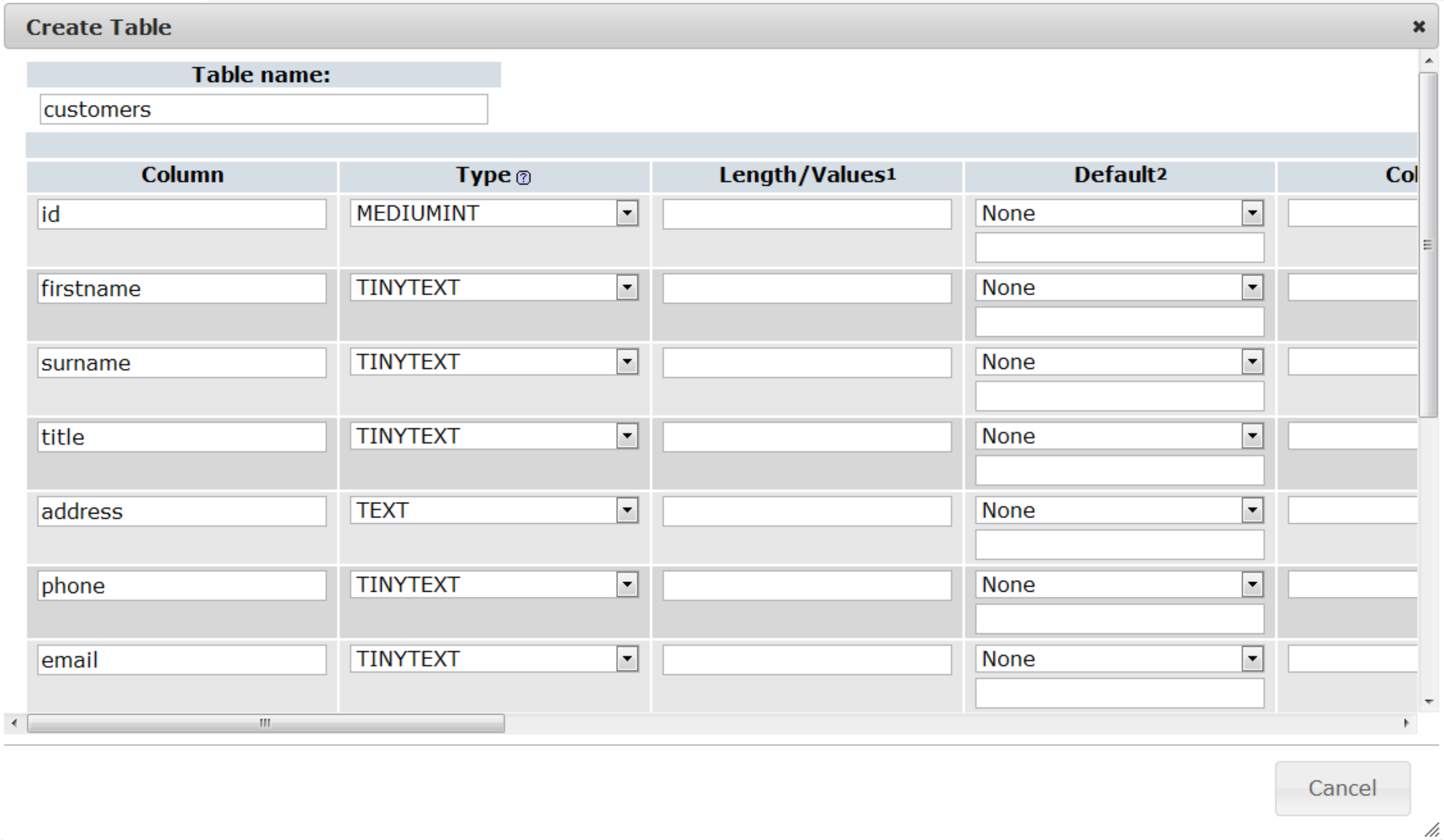
Structure | SQL | Search | Query | Export | Import | Operations | Privileges | Tracking | Designer

No tables found in database.

Create table on database thewebbo_hms

Name: customers Number of columns: 7

Go



Create Table

Table name:

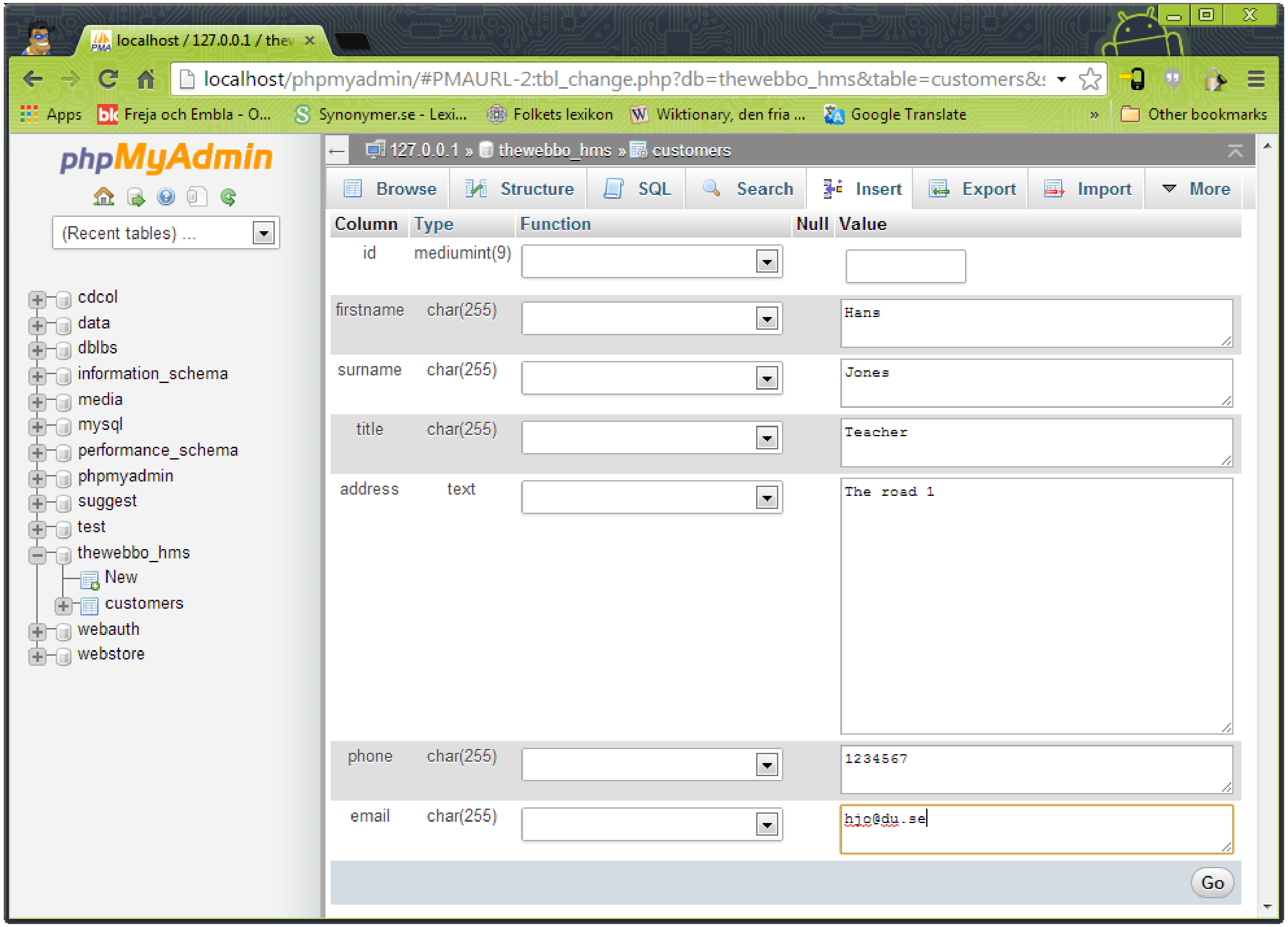
customers

Column	Type	Length/Values1	Default2	Col
id	MEDIUMINT		None	
firstname	TINYTEXT		None	
surname	TINYTEXT		None	
title	TINYTEXT		None	
address	TEXT		None	
phone	TINYTEXT		None	
email	TINYTEXT		None	

Cancel

Hotel management DB 2

- Insert data with the UI



localhost/phpmyadmin/#PMAURL-2:tbl_change.php?db=thewebbo_hms&table=customers&

127.0.0.1 » thewebbo_hms » customers

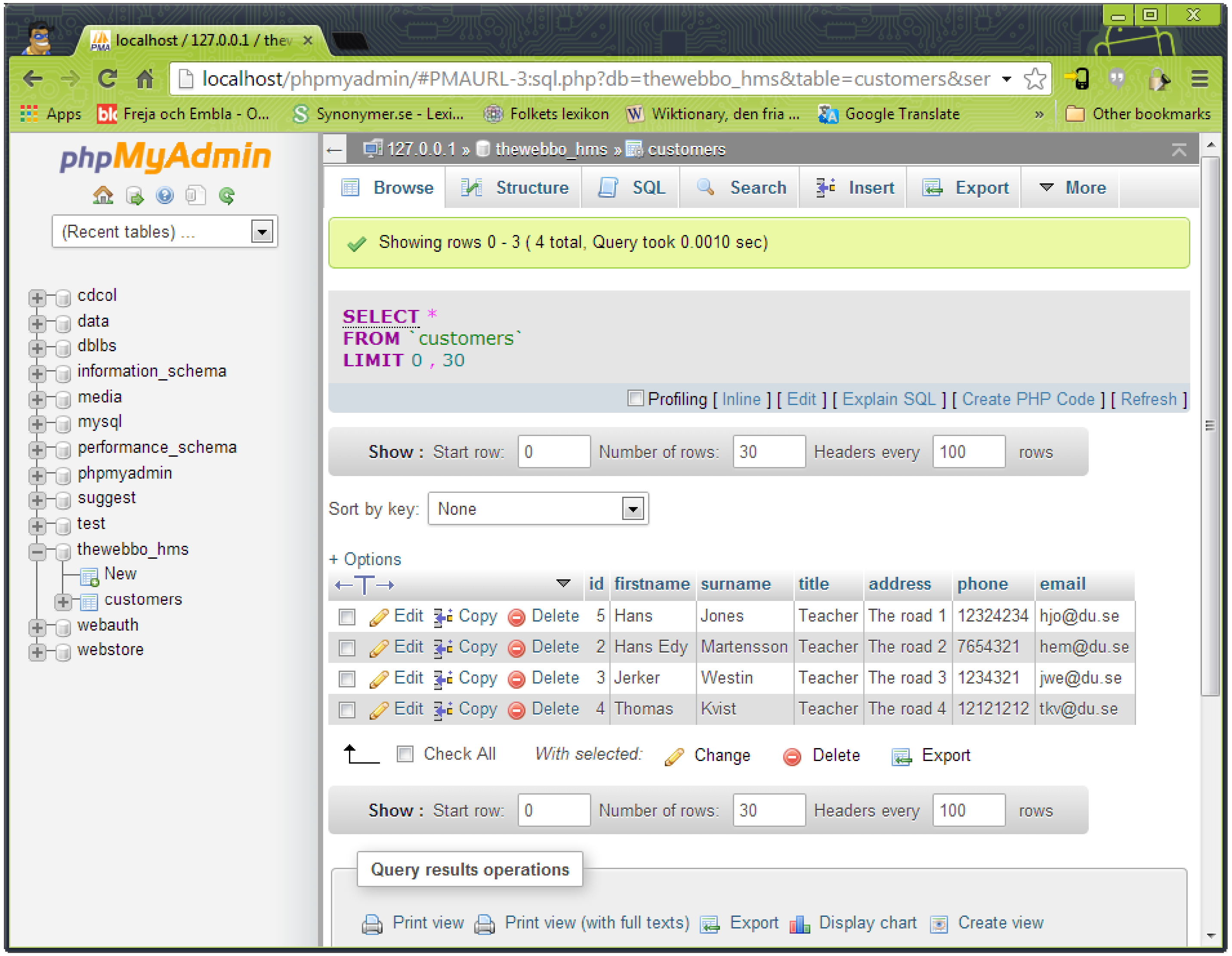
Browse | Structure | SQL | Search | Insert | Export | Import | More

Column	Type	Function	Null	Value
id	mediumint(9)			
firstname	char(255)			Hans
surname	char(255)			Jones
title	char(255)			Teacher
address	text			The road 1
phone	char(255)			1234567
email	char(255)			hjo@du.se

Go

Hotel management DB 3

- Browse and edit the database data with the UI



localhost/phpmyadmin/#PMAURL-3:sql.php?db=thewebbo_hms&table=customers&ser

127.0.0.1 » thewebbo_hms » customers

Browse | Structure | SQL | Search | Insert | Export | More

Showing rows 0 - 3 (4 total, Query took 0.0010 sec)

```
SELECT *
FROM `customers`
LIMIT 0 , 30
```

Profiling [Inline] [Edit] [Explain SQL] [Create PHP Code] [Refresh]

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

Sort by key: None

+ Options

	id	firstname	surname	title	address	phone	email
Edit Copy Delete	5	Hans	Jones	Teacher	The road 1	12324234	hjo@du.se
Edit Copy Delete	2	Hans Edy	Martensson	Teacher	The road 2	7654321	hem@du.se
Edit Copy Delete	3	Jerker	Westin	Teacher	The road 3	1234321	jwe@du.se
Edit Copy Delete	4	Thomas	Kvist	Teacher	The road 4	12121212	tkv@du.se

Check All *With selected:* Change Delete Export

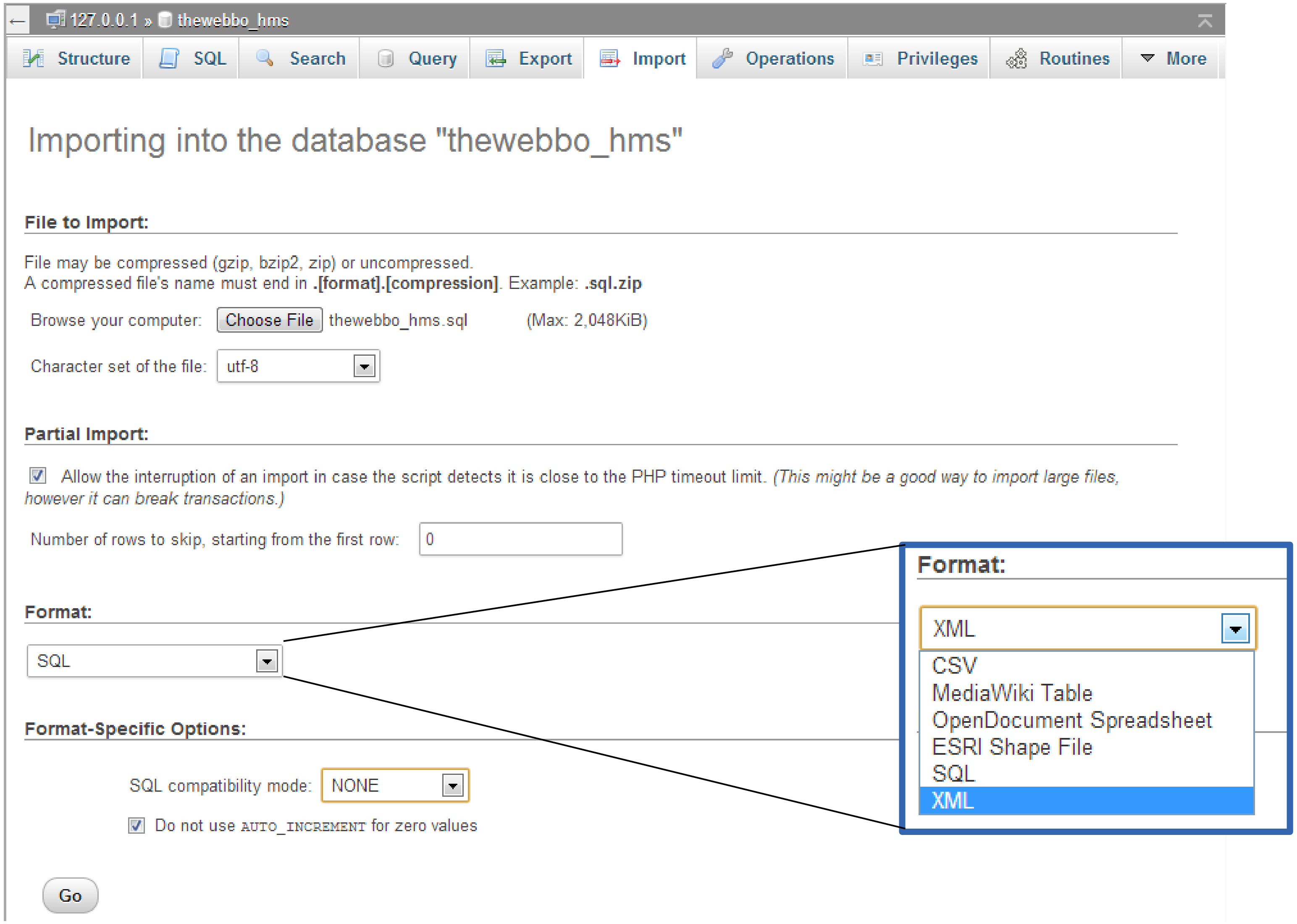
Show : Start row: 0 Number of rows: 30 Headers every 100 rows

Query results operations

Print view Print view (with full texts) Export Display chart Create view

Hotel management DB 4

- Import already created formatted data with the UI



Importing into the database "thewebbo_hms"

File to Import:

File may be compressed (gzip, bzip2, zip) or uncompressed.
A compressed file's name must end in **.[format].[compression]**. Example: **.sql.zip**

Browse your computer: Choose File thewebbo_hms.sql (Max: 2,048KiB)

Character set of the file: utf-8

Partial Import:

☑ Allow the interruption of an import in case the script detects it is close to the PHP timeout limit. *(This might be a good way to import large files, however it can break transactions.)*

Number of rows to skip, starting from the first row: 0

Format:

SQL

Format-Specific Options:

SQL compatibility mode: NONE

☑ Do not use `AUTO_INCREMENT` for zero values

Go

Format:

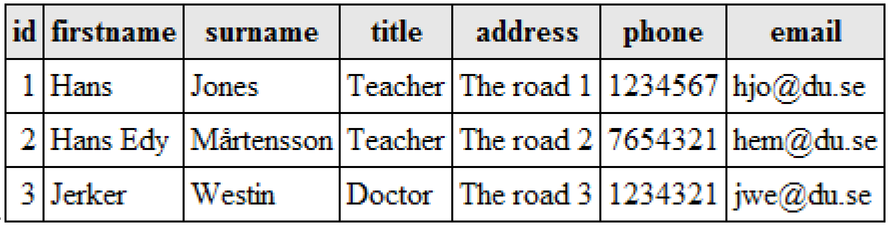
XML

- CSV
- MediaWiki Table
- OpenDocument Spreadsheet
- ESRI Shape File
- SQL
- XML

Do it by hand with SQL

- id? PRIMARY KEY? NOT NULL? AUTO_INCREMENT?

```
CREATE TABLE thewebbo_hms.customers (
     id MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT,
     firstname TINYTEXT NOT NULL,
     surname TINYTEXT NOT NULL,
     title TINYTEXT NOT NULL,
     address TEXT NOT NULL,
     phone TINYTEXT NOT NULL,
     email TINYTEXT NOT NULL,
     PRIMARY KEY (id)
) ENGINE=MyISAM;

INSERT INTO customers (firstname, surname, title, address, phone, email) VALUES
     ('Hans', 'Jones', 'Teacher', 'The road 1', '1234567', 'hjo@du.se'),
     ('Hans Edy', 'Mårtensson', 'Teacher', 'The road 2', '7654321', 'hem@du.se'),
     ('Jerker', 'Westin', 'Doctor', 'The road 3', '1234321', 'jwe@du.se');

SELECT * FROM customers;
```



id	firstname	surname	title	address	phone	email
1	Hans	Jones	Teacher	The road 1	1234567	hjo@du.se
2	Hans Edy	Mårtensson	Teacher	The road 2	7654321	hem@du.se
3	Jerker	Westin	Doctor	The road 3	1234321	jwe@du.se

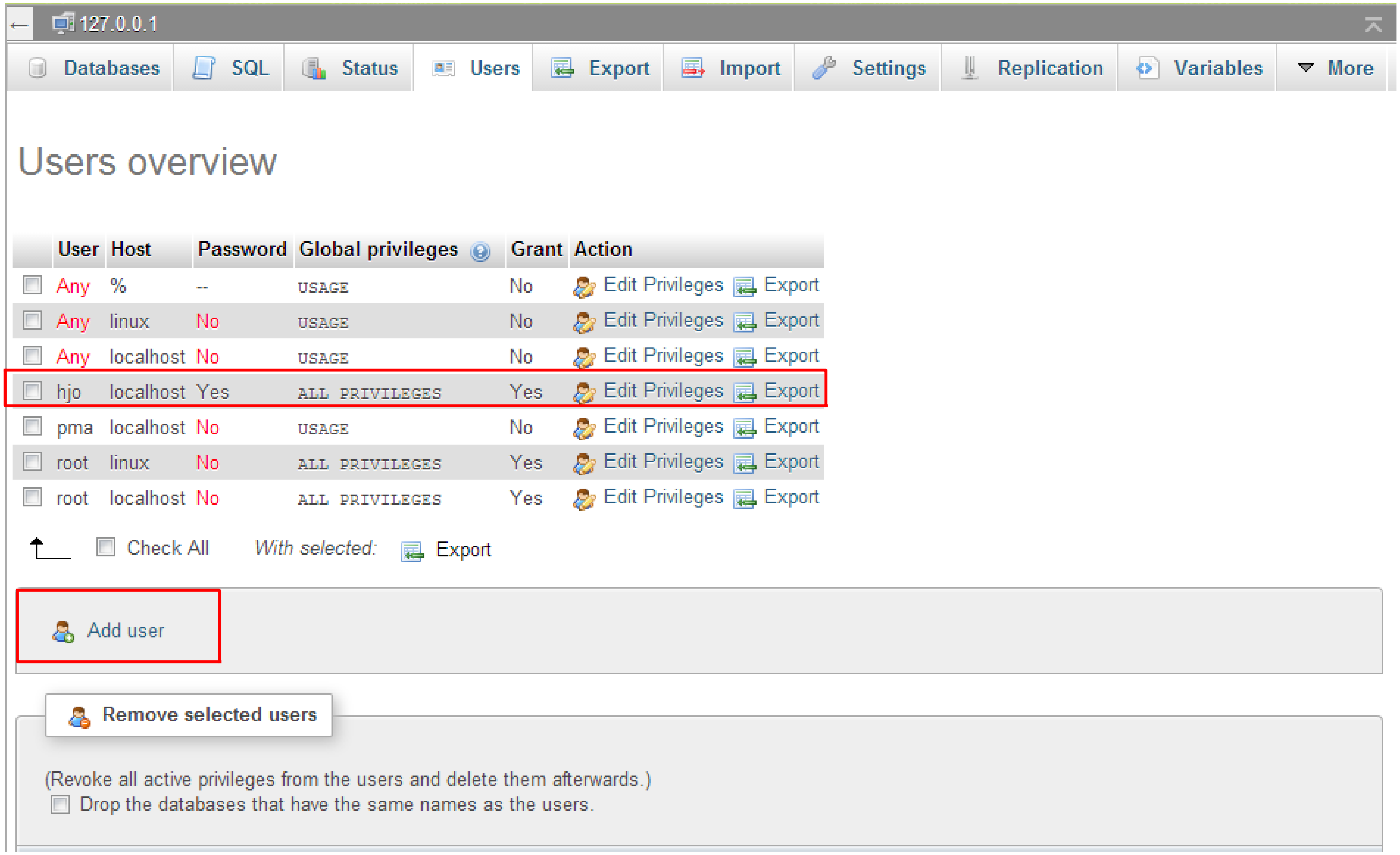
Hotel management DB 5

- Designing the database tables correct from start is very important
  - The data types for the fields (columns)
  - Split for example address in several new columns as address, city, state, county, zip code, country
- MySQL have many table types, MyISAM tables which is the default table type is very simple
  - http://www.sitepoint.com/mysql-myisam-table-pros-con/
- Query the table

```
SELECT firstname FROM customers;
SELECT firstname FROM customers ORDER BY firstname;
SELECT title, firstname, surname FROM customers;
SELECT * FROM customers WHERE id = 2;
SELECT * FROM customers WHERE firstname = "Jerker";
SELECT firstname, surname FROM customers WHERE surname = 'Westin';

DROP TABLE customers;
```

Create a database user 1 and 3

- Using phpMyAdmin and the result after creation



Users overview

	User	Host	Password	Global privileges	Grant	Action
☐	Any	%	--	USAGE	No	Edit Privileges Export
☐	Any	linux	No	USAGE	No	Edit Privileges Export
☐	Any	localhost	No	USAGE	No	Edit Privileges Export
☐	hjo	localhost	Yes	ALL PRIVILEGES	Yes	Edit Privileges Export
☐	pma	localhost	No	USAGE	No	Edit Privileges Export
☐	root	linux	No	ALL PRIVILEGES	Yes	Edit Privileges Export
☐	root	localhost	No	ALL PRIVILEGES	Yes	Edit Privileges Export

☐ Check All *With selected:* Export

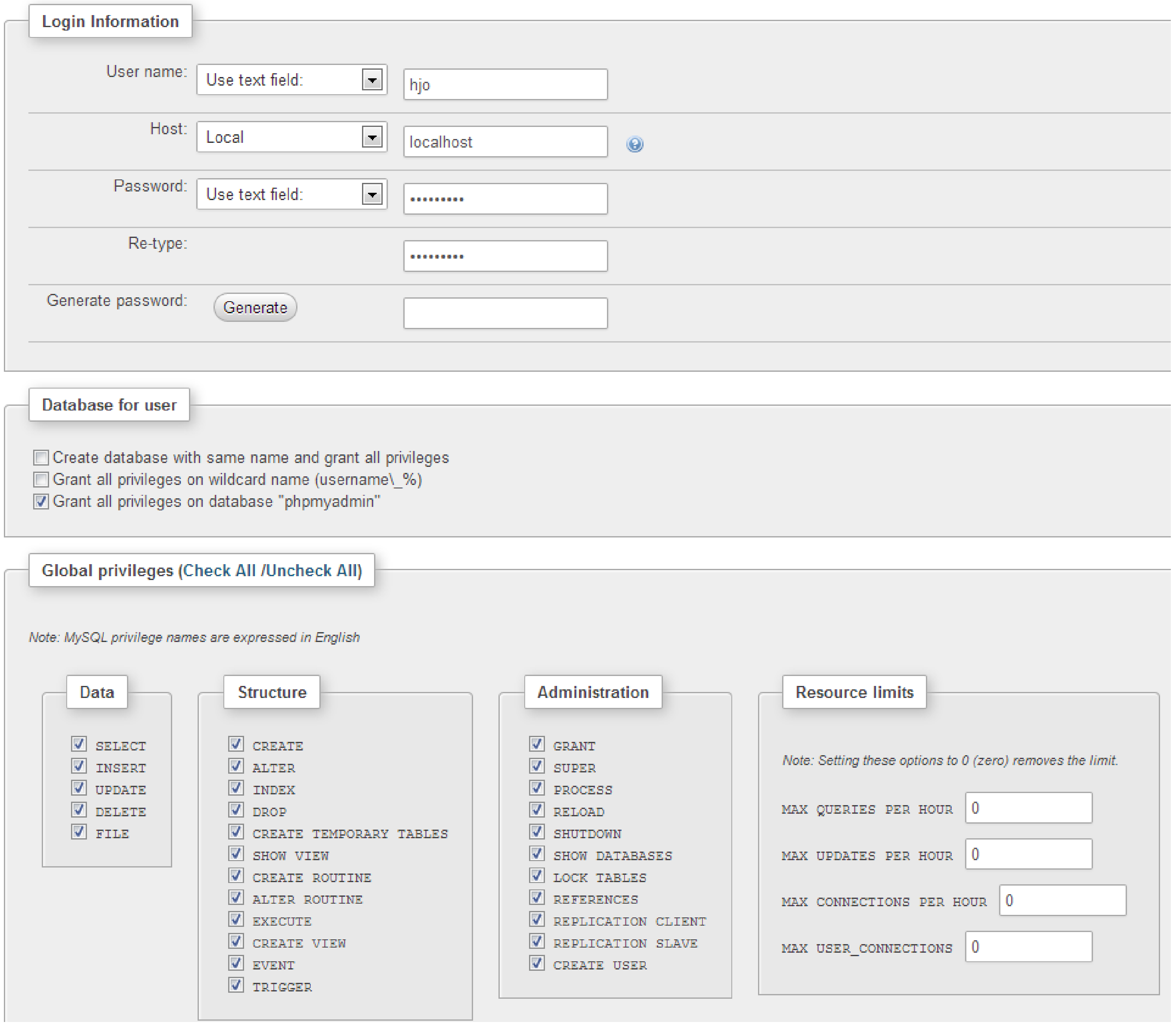
Add user

Remove selected users

(Revoke all active privileges from the users and delete them afterwards.)
☐ Drop the databases that have the same names as the users.

Create a database user 2

- Grant privileges



Login Information

User name:	Use text field:	hjo
Host:	Local	localhost
Password:	Use text field:	•••••••••
Re-type:		•••••••••
Generate password:	Generate	

Database for user

- [ ] Create database with same name and grant all privileges
- [ ] Grant all privileges on wildcard name (username_%)
- [✓] Grant all privileges on database "phpmyadmin"

Global privileges (Check All /Uncheck All)

Note: MySQL privilege names are expressed in English

Data

- [✓] SELECT
- [✓] INSERT
- [✓] UPDATE
- [✓] DELETE
- [✓] FILE

Structure

- [✓] CREATE
- [✓] ALTER
- [✓] INDEX
- [✓] DROP
- [✓] CREATE TEMPORARY TABLES
- [✓] SHOW VIEW
- [✓] CREATE ROUTINE
- [✓] ALTER ROUTINE
- [✓] EXECUTE
- [✓] CREATE VIEW
- [✓] EVENT
- [✓] TRIGGER

Administration

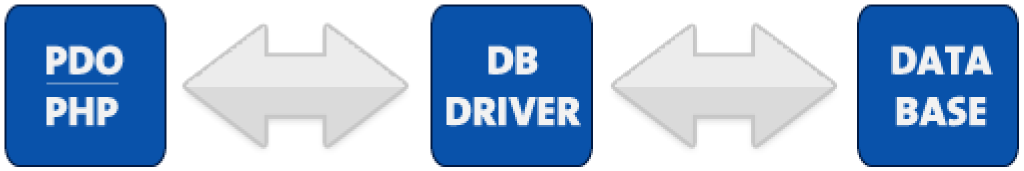
- [✓] GRANT
- [✓] SUPER
- [✓] PROCESS
- [✓] RELOAD
- [✓] SHUTDOWN
- [✓] SHOW DATABASES
- [✓] LOCK TABLES
- [✓] REFERENCES
- [✓] REPLICATION CLIENT
- [✓] REPLICATION SLAVE
- [✓] CREATE USER

Resource limits

Note: Setting these options to 0 (zero) removes the limit.

MAX QUERIES PER HOUR	0
MAX UPDATES PER HOUR	0
MAX CONNECTIONS PER HOUR	0
MAX USER_CONNECTIONS	0

PHP Data Objects (PDO)



- Accessing the database from PHP we can use the MySQL or MySQLi (improved) extensions but better options are avialable
- The PHP Data Objects (PDO) extension defines a lightweight, consistent interface for accessing databases in PHP
- PDO ships with PHP 5.1 and later (supported by XAMPP)
- It will make your database coding in PHP more secure, faster and portable (easier to change and many DBs are supported)
  - `<?php print_r(PDO::getAvailableDrivers()); ?>` returns Array ([0] => mysql [1] => sqlite) in XAMPP
- PDO is object oriented with all the benefits coming with this
- Using prepared statements will help protect you from SQL injection so you do not need any own sanitize functions

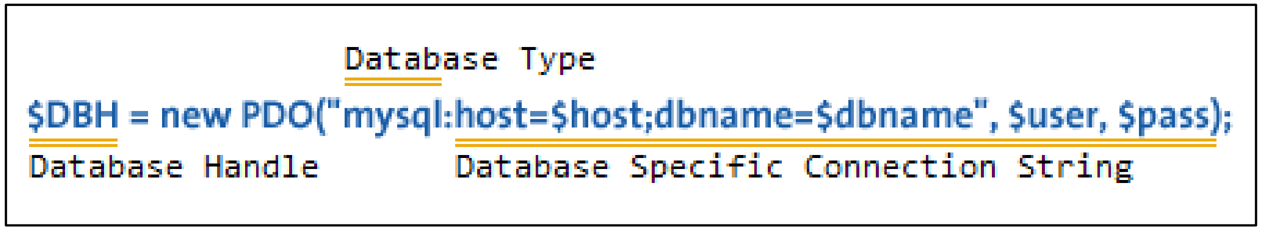
PDO vs. MySQL

- The code below is fairly simple, but it does come with its significant share of downsides
  - **Deprecated**: Though it hasn't been officially deprecated – due to widespread use – in terms of best practice and education, it might as well be
  - **Escaping**: The process of escaping user input is left to the developer – many of which don't understand or know how to sanitize the data
  - **Flexibility**: The API isn't flexible; the code below is tailor-made for working with a MySQL database. What if you switch?

```
<?php
# Connect
mysql_connect('localhost', 'username', 'password') or die('Could not connect: ' . mysql_error());

# Choose a database
mysql_select_db('someDatabase') or die('Could not select database');

# Perform database query
$query = "SELECT * from someTable";
$result = mysql_query($query) or die('Query failed: ' . mysql_error());

# Filter through rows and echo desired information
while ($row = mysql_fetch_object($result)) {
     echo $row->name;
}
?>
```

PDO – connect and close

- Different databases may have slightly different connection methods. Below, the method to connect to some of the most popular databases are shown. You’ll notice that the first three are identical, other then the database type – and then SQLite has its own syntax



```
$DBH = new PDO("mysql:host=$host;dbname=$dbname", $user, $pass);
```

```
<?php
$host = 'utb-mysql.du.se'; $dbname = 'db25'; $user = 'db25'; $pass = 'fGBYZtwY';
try {
     # MS SQL Server and Sybase with PDO_DBLIB
     $DBH = new PDO("mssql:host=$host;dbname=$dbname, $user, $pass");
     $DBH = new PDO("sybase:host=$host;dbname=$dbname, $user, $pass");

     # MySQL with PDO_MYSQL
     $DBH = new PDO("mysql:host=$host;dbname=$dbname", $user, $pass);

     # SQLite Database
     $DBH = new PDO("sqlite:my/database/path/database.db");
}
catch(PDOException $e) {
     echo $e->getMessage();
}
?>
```

```
<?php
# close the db connection
$DBH = null;
?>
```

PDO – exceptions

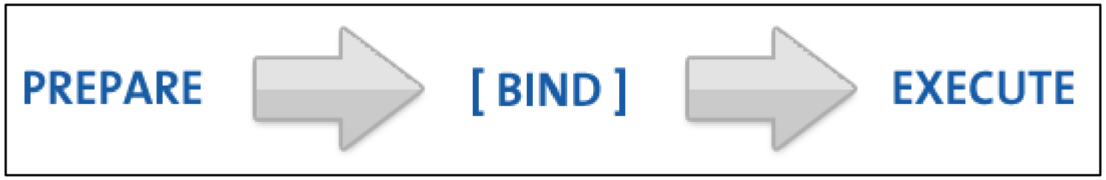
- PDO can use exceptions to handle errors, which means anything you do with PDO should be wrapped in a try/catch block
- You can force PDO into one of 3 error modes by setting the error mode attribute on your newly created database handle

```php
<?php
$DBH->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_SILENT );  # default error mode
$DBH->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING ); # useful for debugging, program will continue to execute
$DBH->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION ); # the mode for most situations, it fires an exception
?>
```

```php
<?php
$host = 'utb-mysql.du.se'; $dbname = 'db25'; $user = 'db25'; $pass = 'fGBYZtwY';
# connect to the database
try {
  $DBH = new PDO("mysql:host=$host;dbname=$dbname", $user, $pass);
  $DBH->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION );  # set error attribute

  # UH-OH! Typed DELECT instead of SELECT!
  $DBH->prepare('DELECT name FROM people');
}
catch(PDOException $e) {
    echo "I'm sorry, Dave. I'm afraid I can't do that.";
    file_put_contents('PDOErrors.txt', $e->getMessage(), FILE_APPEND); # log errors to a file
}
?>
```

PDO – create and update

- Using PDO, create and update is normally a two-step process



```
<?php
# The most basic type of insert, STH means "Statement Handle", no binding here
$STH = $DBH->prepare("INSERT INTO folks ( first_name ) values ( 'Cathy' )");
$STH->execute();
?>
```

- A prepared statement is a precompiled SQL statement that can be executed multiple times by just sending the data to the server
- It has the added advantage of automatically making the data used in the placeholders safe from SQL injection attacks!

```
<?php
# no placeholders - ripe for SQL Injection!
$STH = $DBH->prepare("INSERT INTO folks (name, addr, city) values ($name, $addr, $city)");
# unnamed placeholders
$STH = $DBH->prepare("INSERT INTO folks (name, addr, city) values (?, ?, ?)");
# named placeholders
$STH = $DBH->prepare("INSERT INTO folks (name, addr, city) value (:name, :addr, :city)");
?>
```

PDO - prepared statements 1

- Unnamed placeholders

```
<?php
$STH = $DBH->prepare("INSERT INTO folks (name, addr, city) values (?, ?, ?)");
# assign variables to each place holder, indexed 1-3
$STH->bindParam(1, $name); $STH->bindParam(2, $addr); $STH->bindParam(3, $city);

# insert one row - once the query have been prepared ...
$name = "Daniel";
$addr = "1 Wicked Way";
$city = "Arlington Heights";
$STH->execute();

# ... insert another row with different values – multiple times (looping)
$name = "Steve"
$addr = "5 Circle Drive";
$city = "Schaumburg";
$STH->execute();

# Does this seem a bit unwieldy for statements with a lot of parameters? It is!
# However, if your data is stored in an array, there’s an easy shortcut.
# We do not need to use ->bindParam() - the execute($values) method does this!
# the array data we want to insert must be in the arg. ->execute(argument)
$data = array('Cathy', '9 Dark and Twisty Road', 'Cardiff');
$STH = $DBH->prepare("INSERT INTO folks (name, addr, city) values (?, ?, ?)");
$STH->execute($data);
?>
```

PDO - prepared statements 2

- Named placeholders

```php
<?php
$STH = $DBH->prepare("INSERT INTO folks (name, addr, city) value (:name, :addr, :city)");
# the first argument is the named placeholder name - notice named placeholders always start with a colon
$STH->bindParam(':name', $name); $STH->bindParam(':addr', $addr); $STH->bindParam(':city', $city);

# insert one row - insert as many rows as you want just updating the variables and ->execute()
$name = "Daniel"; $addr = "1 Wicked Way"; $city = "Arlington Heights";
$STH->execute();

# You can use a shortcut here as well, but it works with associative arrays. The data we want to insert
$data = array(':name' => 'Cathy', ':addr' => '9 Dark and Twisty', ':city' => 'Cardiff');
$STH = $DBH->prepare("INSERT INTO folks (name, addr, city) value (:name, :addr, :city)");
# And the array shortcut ->execute(arg)!
$STH->execute($data);

# Another nice feature of named placeholders is the ability to insert objects directly into your
# database, assuming the properties match the named fields - a simple object
class person {
      public $name; public $addr; public $city;
      function __construct($n,$a,$c) {
            $this->name = $n; $this->addr = $a; $this->city = $c;
      }
      # etc ...
}
$cathy = new person('Cathy','9 Dark and Twisty','Cardiff');
# here's the fun part
$STH = $DBH->prepare("INSERT INTO folks (name, addr, city) value (:name, :addr, :city)");
# By casting the object to an array in the execute, the properties are treated as array keys
$STH->execute((array)$cathy);
?>
```

PDO - prepared statements 3

- Update and delete with named placeholders

```php
<?php
// update using named place holders
$id = 5;
$name = "Joe the Plumber";
try {
      $DBH = new PDO('mysql:host=localhost;dbname=someDatabase', $username, $password);
      $DBH->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

      $STH = $DBH->prepare('UPDATE someTable SET name = :name WHERE id = :id');
      $result = $STH->execute(array(':id' => $id, ':name' => $name));
      echo $STH->rowCount(), " - ", $result;
}
catch(PDOException $e) {
      echo 'Error: ' . $e->getMessage();
}
// delete using named place holders and the bindParam method
$id = 5;
try {
      $DBH = new PDO('mysql:host=localhost;dbname=someDatabase', $username, $password);
      $DBH->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

      $STH = $DBH->prepare('DELETE FROM someTable WHERE id = :id');
      $STH->bindParam(':id', $id);
      $result = $STH->execute();
      echo $STH->rowCount(), " - ", $result;
}
catch(PDOException $e) {
      echo 'Error: ' . $e->getMessage();
}
?>
```

PDO – retrieve data 1

OBJECTS
PDO
PHP
ARRAYS

- Data is obtained via the ->fetch(), a method of your statement handle. Before calling fetch, it's best to tell PDO how you'd like the data to be fetched
  - There's two core ways to fetch: **query()** and **execute()**
- The most common options which cover most situations are
  - PDO::FETCH_ASSOC: returns an array indexed by column name
  - PDO::FETCH_CLASS: Assigns the values of your columns to properties of the named class. It will create the properties if matching properties do not exist
  - PDO::FETCH_OBJ: returns an anonymous object with property names that correspond to the column names

```
<?php
# In order to set the fetch method, the following syntax is used
$STH->setFetchMode(PDO::FETCH_ASSOC);
?>
```

PDO – retrieve data 2

- FETCH_ASSOC
  - This fetch type creates an associative array, indexed by column name

```php
<?php
# using the shortcut ->query() method here since there are no variable values in the select statement
$STH = $DBH->query('SELECT name, addr, city from folks');

# setting the fetch mode PDO::FETCH_ASSOC - which also is the default fetch mode if not set
$STH->setFetchMode(PDO::FETCH_ASSOC);

# showing the results
while($row = $STH->fetch()) {
    echo $row['name'] . "\n";  echo $row['addr'] . "\n";  echo $row['city'] . "\n";
}
?>
```

- FETCH_OBJ
  - Creates an object of std class for each row of fetched data

```php
<?php
# creating the statement manually escaping the users data with the PDO::quoute() method making it safer
$STH = $DBH->query('SELECT name, addr, city from folks where name = ' . $DBH->quote($name));

# You can also set the fetch mode directly within the ->fetch() method call as well
while($row = $STH->fetch(PDO::FETCH_OBJ)) {
     echo $row->name . "\n"; echo $row->addr . "\n"; echo $row->city . "\n";
}
?>
```

PDO – retrieve data 3

- It's strongly advised that you use prepared statements in fetch as well
- However if your SQL queries are NOT dependent upon form data, the query method is OK to use
- In this example, we're using the prepare method to prepare the query before the user's data has been attached
- With this technique, SQL injection is virtually impossible, because the data doesn't ever get inserted into the SQL query, itself. Notice that, instead, we use named parameters (:name) to specify placeholders.

```php
<?php
# Best Practice for fetch - The Prepared Statements Method
$name = "Daniel";
try {
      $DBH = new PDO('mysql:host=localhost;dbname=myDatabase', $username, $password);
      $DBH->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

      $STH = $DBH->prepare('SELECT * FROM myTable WHERE name = :name');
      $data = array(':name' => $name);
      $STH->execute($data);

      while($row = $STH->fetch()) {
            print_r($row); # displays information about a variable in a way that's readable by humans
      }
}
catch(PDOException $e) {
      echo 'ERROR: ' . $e->getMessage();
}
?>
```

PDO – retrieve data 4

- One of the neatest aspects of PDO is that it gives the ability to map the query results to a class instance or object

```php
<?php
class User {
	public $first_name, $last_name;
	function __construct($ln = '') {
		$this->first_name = preg_replace('/[a-z]/', 'x', $this->first_name);
		$this->last_name = $ln;
	}
	public function full_name(){
		return $this->first_name . ' ' . $this->last_name;
	}
}
try {
	$DBH = new PDO('mysql:host=localhost;dbname=someDatabase', $username, $password);
	$DBH->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

	$STH = $DBH->query('SELECT * FROM someTable');

	# Map results to a User object
	$STH->setFetchMode(PDO::FETCH_CLASS, 'User'); # calls the constructor after the data is assigned in PDO
	# calls the constructor before the data is assigned in PDO
	$STH->setFetchMode(PDO::FETCH_CLASS | PDO::FETCH_PROPS_LATE, 'User');

	while($user = $STH->fetch()) {
		# Call our custom full_name method
		echo $user->full_name();
	}
}
catch(PDOException $e) {
	echo 'Error: ' . $e->getMessage();
}
?>
```

```php
# if you need you can pass arguments to the constructor
# when fetching data into objects with PDO:
$STH->setFetchMode(PDO::FETCH_CLASS, 'User', array('stuff'));
# If you need to pass different data to the constructor for each
# object you can set the fetch mode inside the fetch method:
$i = 0;
while($rowObj = $STH->fetch(PDO::FETCH_CLASS, 'User', array($i))) {
	// do stuff
	$i++;
}
```

PDO – helpful methods

- PDO is a large extension, here are a few more methods to do basic things

```php
<?php
file_put_contents($filename, $data, $flags); # ensure that you got permission to write the file, chmod xxx

# The ->lastInsertId() method is always called on the database handle, not statement handle,
# and will return the auto incremented id of the last inserted row by that connection.
$DBH->lastInsertId();

# The ->exec() method is used for operations that can not return data other then the
# affected rows as database schema changes etc. Here are two examples of using the exec method.
$DBH->exec('DROP TABLE folks');
$DBH->exec("SET time_zone = '-8:00'");

# The ->quote() method quotes strings so they are safe to use in queries.
# This is your fallback if you’re not using prepared statements.
$safe = $DBH->quote($unsafe);

# The ->rowCount() method returns an integer indicating the number of rows affected by an operation
$rows_affected = $STH->rowCount();

# If the rowCount method does not work you can get the numbers of rows with
# (this is currently the case with SQLite, which need this fix)
$sql = "SELECT COUNT(*) FROM folks";
if ($STH = $DBH->query($sql)) {
      if ($STH->fetchColumn() > 0) {     # check the row count
            # issue a real select here, because there's data!
      }
      else {
            echo "No rows matched the query.";
      }
}
?>
```

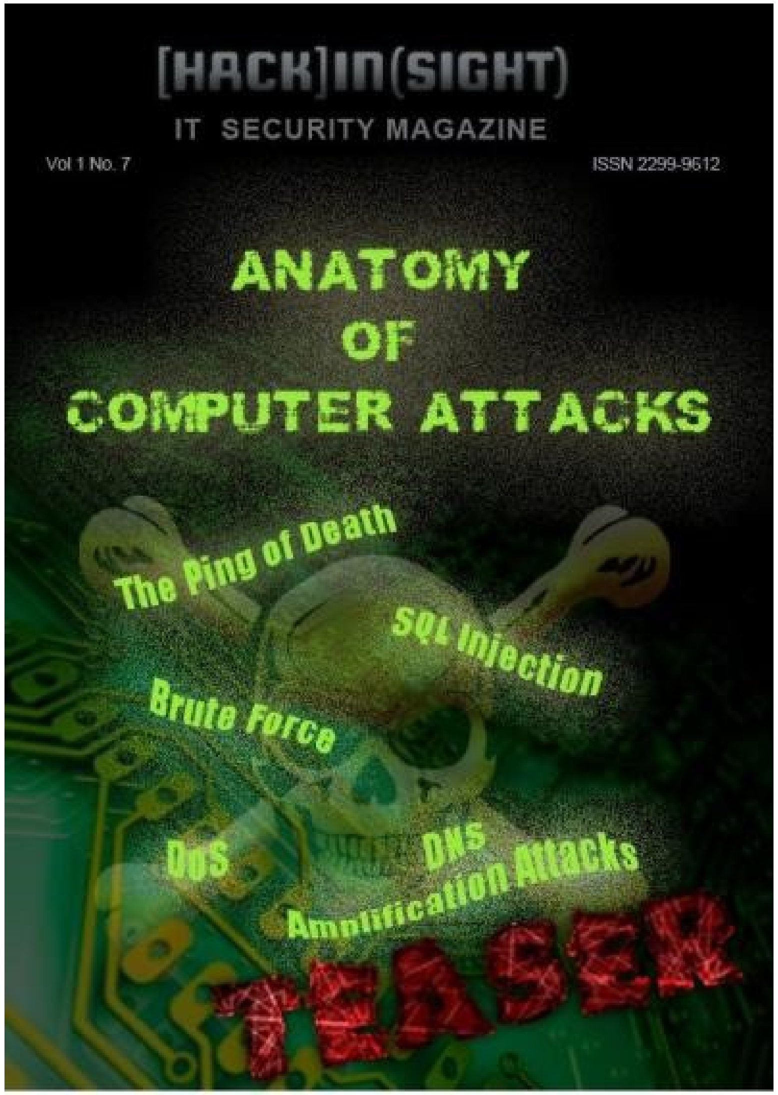
```php
<?php
// PDO supports transactions as well
$DBH->beginTransaction();
// modify db ...
// either commit or rollback!
if($someResultCheck == true)
      $DBH->commit();
else
      $DBH>rollback();
?>
```

PDO CRUD resources

- Introduction to PHP PDO
  - http://www.phpro.org/tutorials/Introduction-to-PHP-PDO.html
- Why you should be using PHP’s PDO for Database Access
  - The article contains some errors so read the comments!
  - http://net.tutsplus.com/tutorials/php/why-you-should-be-using-phps-pdo-for-database-access/
- PHP Database Access: Are You Doing It Correctly?
  - http://net.tutsplus.com/tutorials/php/php-database-access-are-you-doing-it-correctly/
- PHP Data Objects documentation
  - http://www.php.net/manual/en/book.pdo.php

SQL injection and why PDO!

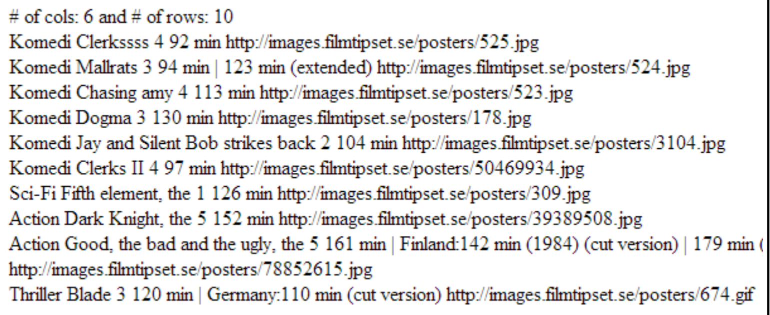
- SQL injection article
  - http://users.du.se/~hjo/cs/dt1040/docs/hi_07_2013_teaser.pdf
- PDO vs. MySQLi
  - PDO is better on all points!
  - http://net.tutsplus.com/tutorials/php/pdo-vs-mysqli-which-should-you-use/
- PDO and MySQLi protects against first order SQL injection if they are correctly used
  - http://download.oracle.com/oll/tutorials/SQLInjection/html/lesson1/les01_tm_attacks.htm



pdo-test.php

- Open DB server at school and print out the filmer table

```
<?php
echo "<html><body>";
$dsn = 'mysql:host=utb-mysql.du.se;dbname=db25;charset=UTF8';
$dbuser = 'db25';
$dbpass = 'fGBYZtwY';
$DBH = null;
$filmer = "filmer";
try{
	# a DB Handler to manage the database connection
	$DBH = new PDO($dsn, $dbuser, $dbpass);
	# SQL query
	$sql = 'SELECT * FROM ' . $filmer;
	# STH means "Statement Handle"
	$STH = $DBH->query($sql);
	# setting the fetch mode (array indexed by column name)
	$STH->setFetchMode(PDO::FETCH_ASSOC);

	echo "# of cols: ", $STH->columnCount(), " and # of rows: ", $STH->rowCount(), "<br />";

	while($row = $STH->fetch()) {
	    echo "{$row['genre']} {$row['titel']} {$row['betyg']} {$row['langd']} {$row['bild']}<br />";
	}
	# close the connection
	$DBH = null;
}
catch (PDOException $e){
	echo '<b>PDOException: </b>', $e->getMessage();
	die();
}
echo "</body></html>";
?>
```

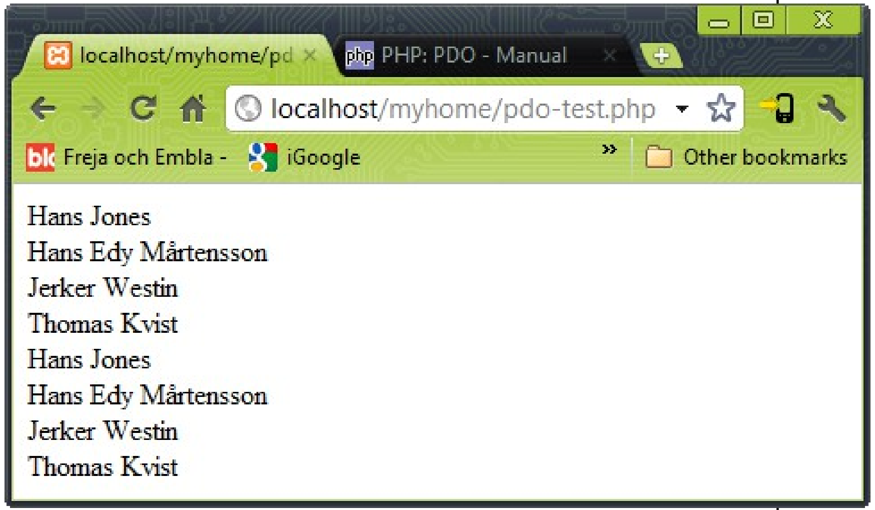


```
# of cols: 6 and # of rows: 10
Komedi Clerkssss 4 92 min http://images.filmtipset.se/posters/525.jpg
Komedi Mallrats 3 94 min | 123 min (extended) http://images.filmtipset.se/posters/524.jpg
Komedi Chasing amy 4 113 min http://images.filmtipset.se/posters/523.jpg
Komedi Dogma 3 130 min http://images.filmtipset.se/posters/178.jpg
Komedi Jay and Silent Bob strikes back 2 104 min http://images.filmtipset.se/posters/3104.jpg
Komedi Clerks II 4 97 min http://images.filmtipset.se/posters/50469934.jpg
Sci-Fi Fifth element, the 1 126 min http://images.filmtipset.se/posters/309.jpg
Action Dark Knight, the 5 152 min http://images.filmtipset.se/posters/39389508.jpg
Action Good, the bad and the ugly, the 5 161 min | Finland:142 min (1984) (cut version) | 179 min (
http://images.filmtipset.se/posters/78852615.jpg
Thriller Blade 3 120 min | Germany:110 min (cut version) http://images.filmtipset.se/posters/674.gif
```

List customers - thewebbo_hms

- PDO read example

```php
<?php
echo "<html><body>";
$dbtype = "mysql"; // PDO configuration
$dbhost = "localhost";
$dbname = "thewebbo_hms";
$dbuser = "hjo";
$dbpass = "abc123xyz";
$charset = "UTF8";
$dsn = "$dbtype:host=$dbhost;dbname=$dbname;charset=$charset";

$query = "select firstname, surname from customers";

# MySQL with PDO_MYSQL
$DBH = $DBH = new PDO($dsn, $dbuser, $dbpass);

# using the shortcut ->query() method here since there are no variable
# values in the select statement.
$STH = $DBH->query($query);

# setting the fetch mode (array indexed by column name)
$STH->setFetchMode(PDO::FETCH_ASSOC);

while($row = $STH->fetch()) {
    echo $row['firstname'] . " " . $row['surname'] . "<br />";
}
################## Do it again ###########################
$STH = $DBH->query($query);

# setting fetch mode (object with names by column name)
$STH->setFetchMode(PDO::FETCH_OBJ);

# showing the results
while($row = $STH->fetch()) {
      echo $row->firstname . " " . $row->surname . "<br />";
}
echo "</body></html>";
?>
```



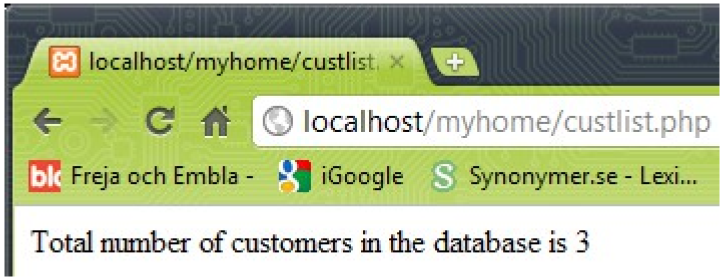
```
Hans Jones
Hans Edy Mårtensson
Jerker Westin
Thomas Kvist
Hans Jones
Hans Edy Mårtensson
Jerker Westin
Thomas Kvist
```

Counting rows - thewebbo_hms

- Try if it works with some simple script

```
<?php
include "utils.php";
# Start the page properly, found in utils.php
printHeader();
try{
	if($DBH == null)
		$DBH = new PDO($dsn, $dbuser, $dbpass);
	# using the shortcut ->query() method here since there
	# are no variable values in the select statement
	$query = 'SELECT * FROM customers';
	$STH = $DBH->query($query);
	$total = $STH->rowCount();

	if ($total == 0) {
		echo "Sorry, no matches found!";
	}
	else {
		echo "Total number of customers in the database is " . $total;
	}
	# close the connection
	$DBH = null;
}
catch (PDOException $e){
	echo '<b>PDOException: </b>',$e->getMessage();
	die();
}
# End the page properly, found in utils.php
printFooter();
?>
```

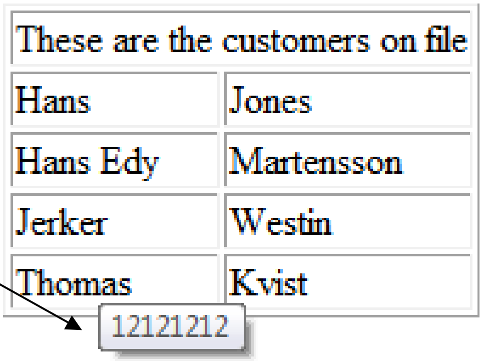
```
# If the rowCount method does not work you can get
# the numbers of row with
$sql = "SELECT COUNT(*) FROM customers";
$STH = $DBH->query($sql);
if($STH->fetchColumn() > 0) { # check the row count
# ...
}
```



Total number of customers in the database is 3

Reading data – make it nice - thewebbo_hms

```php
<?php
include "utils.php";
printHeader(); # Start the page properly, found in utils.php
try{
      # a DB Handler to manage the database connection
      if($DBH == null)
            $DBH = new PDO($dsn, $dbuser, $dbpass);

      $sql = "select firstname, surname, phone from customers";
      # STH means "Statement Handle"
      $STH = $DBH->query($sql);
      # setting the fetch mode
      $STH->setFetchMode(PDO::FETCH_OBJ);

      #echo "# of cols: " . $STH->columnCount() . " and # of rows: " . $STH->rowCount() . "<br />";
      echo "<table border='1'><tr>";
      echo "<td colspan='2'>These are the customers on file</td></tr>";

      while ($row = $STH->fetch()) {
            echo "<tr><td title={$row->phone}>{$row->firstname}</td>";
            echo "<td>{$row->surname}</td></tr>";
      }
      echo "</table>";
      # close the connection
      $DBH = null;
}
catch (PDOException $e){
      echo '<b>PDOException: </b>',$e->getMessage();
      die();
}
printFooter(); # End the page properly, found in utils.php
?>
```

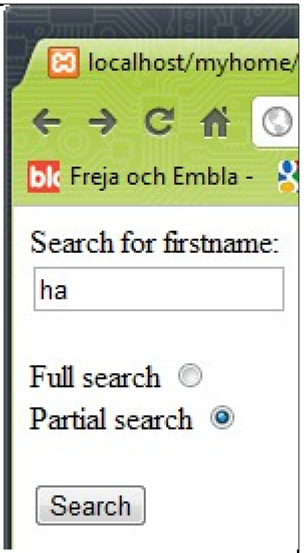


These are the customers on file	
Hans	Jones
Hans Edy	Martensson
Jerker	Westin
Thomas	Kvist

12121212

Searching a table 1 - thewebbo_hms

```php
<?php
include "utils.php";
# SEARCH.PHP - search for firstname
# Start the page properly, SEARCH.PHP - search for firstname
printHeader();

# Check whether the searchtype radio button has been set
# If not set, display the search form.
if (!isset($_POST["searchtype"])) {
      echo "<form method='POST' action='search.php'>";
      echo "Search for firstname:<br>";
      # using html5 input type search
      echo "<input type='search' name='searchtext' size='15'
            placeholder='search' results='5'' autosave='saved-searches'>";
      echo "<br><br>";
      echo "Full search ";
      echo"<input type='radio' value='FULL' checked name='searchtype'><br>";
      echo "Partial search ";
      echo "<input type='radio' name='searchtype' value='PARTIAL'>";
      echo "<br><br>";
      echo "<input type='submit' value='Search' name='submit'>";
      echo "</form>";
} # if
else {  # Searchtype was set, so retrieve form data and do the search
      $searchtext = $_POST["searchtext"]; # Retrieve from the form
      $searchtype = $_POST["searchtype"]; # Retrieve from the form
      $searchtext_san = sanitize_form_text($searchtext); # Prevents SQL injections!

      # Now connect to the database
      try{
            if($DBH == null)
                  $DBH = new PDO($dsn, $dbuser, $dbpass);
```



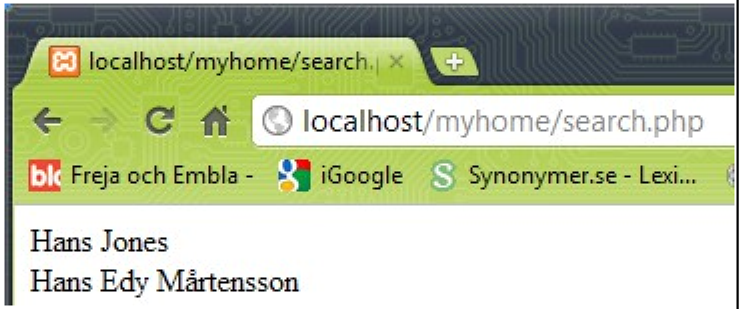
Search for firstname:

ha

Full search ○
Partial search ●

Searching a table 2 - thewebbo_hms

```
            # Construct the appropriate querys
            $sql = "select firstname, surname from customers where firstname";

            if ($searchtype == "FULL"){
                  $sql .= " = :searchtext_san";
                  $STH = $DBH->prepare($sql);
                  $STH->execute(array(':searchtext_san' => $searchtext_san));
            }

            if ($searchtype == "PARTIAL"){
                  $sql .= " LIKE :searchtext_san";
                  $STH = $DBH->prepare($sql);
                  $STH->execute(array(':searchtext_san' => '%'.$searchtext_san.'%'));
            }
            $STH->setFetchMode(PDO::FETCH_ASSOC);  # setting the fetch mode

            $total = $STH->rowCount();
            if ($total == 0){
                  echo "Sorry, no matches found!";
            }
            if ($total > 0){
                  while ($row = $STH->fetch()){
                        echo "{$row["firstname"]} {$row["surname"]}<br>";
                  } # while
            } # if matches found
            # close the connection
            $DBH = null;
      }
      catch (PDOException $e){
            echo '<b>PDOException: </b>',$e->getMessage();
            die();
      }
} # else
PrintFooter(); # End the page properly
?>
```



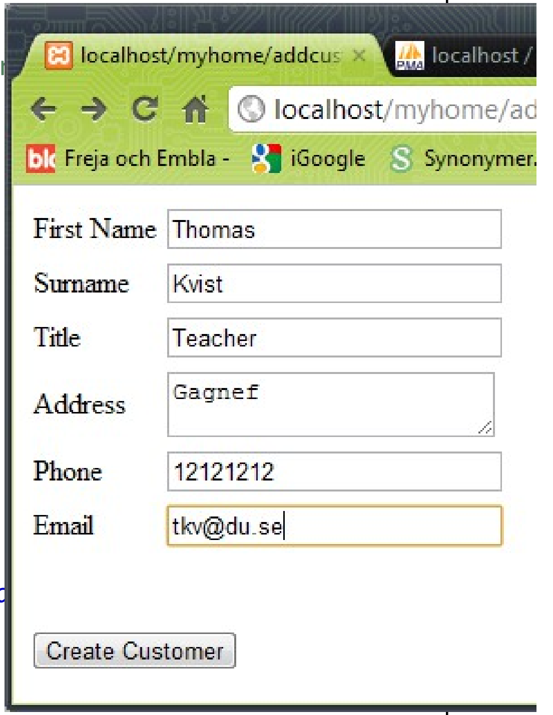
Hans Jones
Hans Edy Mårtensson

Preventing SQL Injection Attacks before PDO

- In the previous example we did something like: select firstname, surname from customers where surname = 'Smith'
  - But what if the visitor enters some search text as follows: Smith' or surname != 'Smith
  - We end up with: select firstname, surname from customers where surname = 'Smith' or surname != 'Smith'
  - In other words, it will return the entire contents of the table!
- Consider what happens if the following is entered as a surname: Smith' or surname != 'Smith; delete from customers
  - The semicolon is the standard character in MySQL for separating multiple commands on a single line. So now, after your program searches for the entered surname, it will then delete the entire contents of your customer database!
- Note that we can enter characters in HEX code as well %3B = **;** which means that we must block the **%** too
- Attackers have sophisticated tools that automatically look for such errors on web sites and try to exploit them!

Adding data 1 - thewebbo_hms

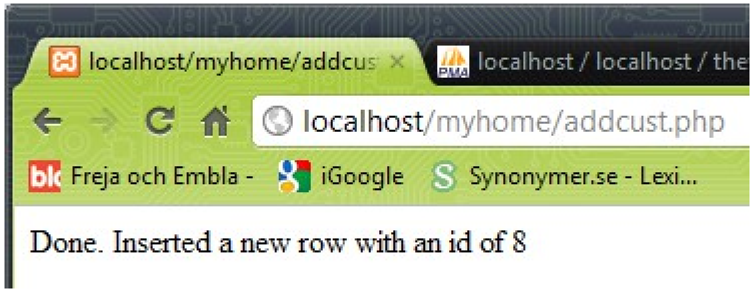
```
<?php
include "utils.php";
# ADDCUST.PHP - Add a new customer to the customers table
# Check whether there's a surname specified. If not, then just display the fo
if(!isset($_POST["tb_surname"])) { # if no surname specified
      printHeader();
      echo "<form method='POST' action='addcust.php'>";
      echo "<table><tr>";
      echo "<td>First Name</td>";
      echo "<td><input type='text' name='tb_firstname' size='25'></td>";
      echo "</tr><tr>";
      echo "<td>Surname</td>";
      echo "<td><input type='text' name='tb_surname' size='25'></td>";
      echo "</tr><tr>";
      echo "<td>Title</td>";
      echo "<td><input type='text' name='tb_title' size='25'></td>";
      echo "</tr><tr>";
      echo "<td>Address</td>";
      echo "<td><textarea rows='2' name='ta_address' cols='20'></textarea></td
      echo "</tr><tr>";
      echo "<td>Phone</td>";
      echo "<td><input type='text' name='tb_phone' size='25'></td>";
      echo "</tr><tr>";
      echo "<td>Email</td>";
      echo "<td><input type='text' name='tb_email' size='25'></td>";
      echo "</tr></table>";
      echo "<br>";
      echo "<input type='submit' value='Create Customer' name='button'>";
      echo "</form>";
      printFooter();
} # if no surname specified
else {
# a surname was specified so create a new record and retrieve the data from the form
```



First Name	Thomas
Surname	Kvist
Title	Teacher
Address	Gagnef
Phone	12121212
Email	tkv@du.se

Adding data 2 - thewebbo_hms

```
      $title = $_POST["tb_title"];
      $firstname = $_POST["tb_firstname"];
      $surname = $_POST["tb_surname"];
      $address = $_POST["ta_address"];
      $phone = $_POST["tb_phone"];
      $email = $_POST["tb_email"];
      # Now connect to the database
      try{
            # a DB Handler to manage the database connection
            if($DBH == null)
                  $DBH = new PDO($dsn, $dbuser, $dbpass);
            # Now construct the query to create the new record with named placeholders
            $sql = "insert into customers ";
            $sql .= "(firstname, surname, title, address, phone, email) ";
            $sql .= "values(:firstname,:surname,:title,:address,:phone,:email)";
            #echo $sql;
            # STH means "Statement Handle"
            $STH = $DBH->prepare($sql);
            $result = $STH->execute(array(':firstname'=>$firstname,':surname'=>$surname,
                        ':title'=>$title,':address'=>$address,':phone'=>$phone,':email'=>$email));
            $m = "Done! Inserted $result new row. To return to the ";
            $m .= "home screen click <a href='addcust.php'>here.</a>";
            show_message($m);
            # close the connection
            $DBH = null;
      }
      catch (PDOException $e){
            echo '<b>PDOException: </b>',$e->getMessage();
            die();
      }
      printFooter();
} # else
?>
```



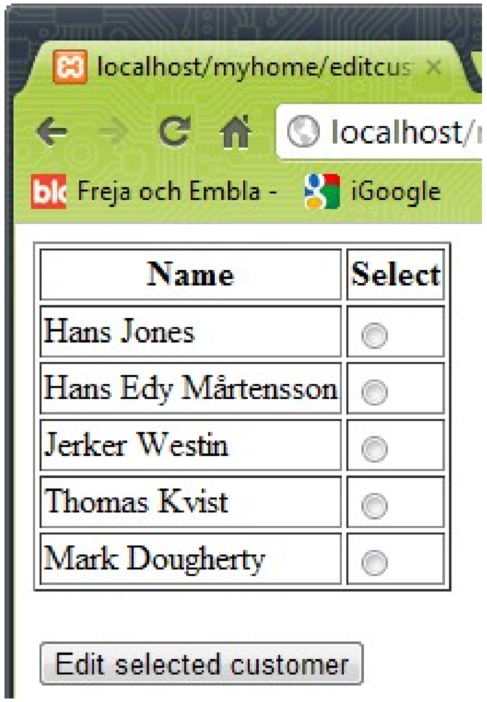
Done. Inserted a new row with an id of 8

Editing a data record

- Remembering that the Web is a stateless system, where each page has no idea what came before it, a parameter is used on the URL to maintain state
- Then, we use $_GET["action"] to retrieve the value of the "action" parameter from the URL which tells us what to do next
- The name of the radio button is id2edit. The value of each radio button is the database id number of the customer
  - This is standard practice, and a very common way of using radio buttons (or dropdown boxes) to select a database record
  - The "action" for the form is set to: **editcust.php?action=show_record**
- When we detect that $_GET["action"] is "show_record", we know that we need to display the customer record form and that we also need to pre-load it with the data for the specified customer as:
  - <input fieldtype="text" name="firstname" value="Robert" size="25">
- Next the action for this editing form is set to: **editcust.php?action=store_record**
  - This time we retrieve the edited values from the form and update the database accordingly. **Note the usage of the hidden field id2edit**

Edit a data record 1 - thewebbo_hms

```php
<?php
include "utils.php";
# EDITCUST.PHP - Allow the user to edit a customer record
# Connect to our database. We'll need this regardless.
$db_host = "localhost"; $db_database = "thewebbo_hms";
try{
        # a DB Handler to manage the database connection
        if($DBH == null)
                $DBH = new PDO($dsn, $dbuser, $dbpass);
}
catch (PDOException $e){
        echo '<b>PDOException: </b>',$e->getMessage();
        die();
}
# Now check the action parameter from the URL to see what we need to do
$action = empty($_GET['action']) ? "" : $_GET['action'];
if ($action == "") { # No action specified so show the home page
        try{
                $sql = "select id, firstname, surname from customers";
                # STH means "Statement Handle"
                $STH = $DBH->query($sql);
                # setting the fetch mode
                $STH->setFetchMode(PDO::FETCH_ASSOC);
                printHeader();
                echo "<form method='post' action='editcust.php?action=show_record'>";
                echo "<table border=1>";
                # Create the table top row
                echo "<tr><th>Name</th><th>Select</th></tr>";
                while ($row = $STH->fetch()) {
                        echo "<tr><td>{$row["firstname"]} {$row["surname"]}</td>";
                        echo "<td>";
                        echo "<input type='radio' value='{$row["id"]}' name='id2edit'>";
                        echo "</td></tr>";
                }
                echo "</table>";
                echo "<br>";
                echo "<input type='submit' value='Edit selected customer' name='button'>";
                echo "</form>";
                $DBH = null; # close the connection
        }
        catch (PDOException $e){
                echo '<b>PDOException: </b>',$e->getMessage();
                die();
        }
        printFooter();
} # action = ""
```

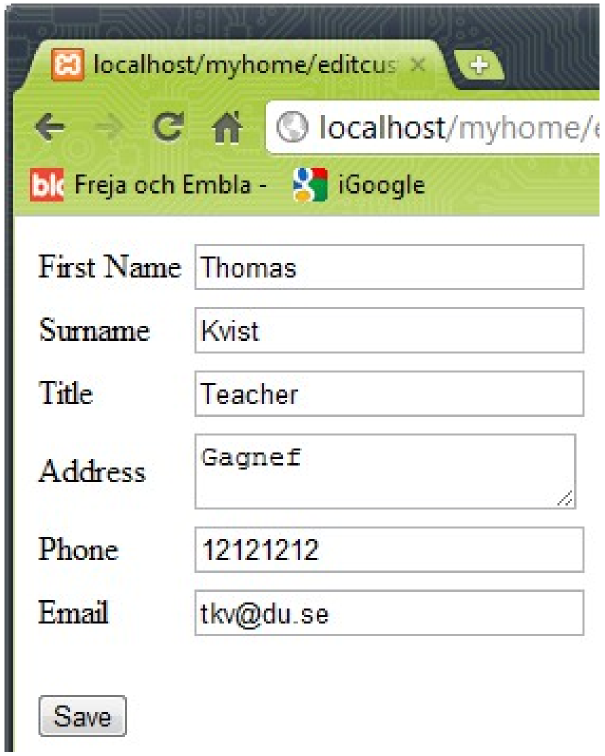


Name	Select
Hans Jones	○
Hans Edy Mårtensson	○
Jerker Westin	○
Thomas Kvist	○
Mark Dougherty	○

Edit selected customer

Edit a data record 2 - thewebbo_hms

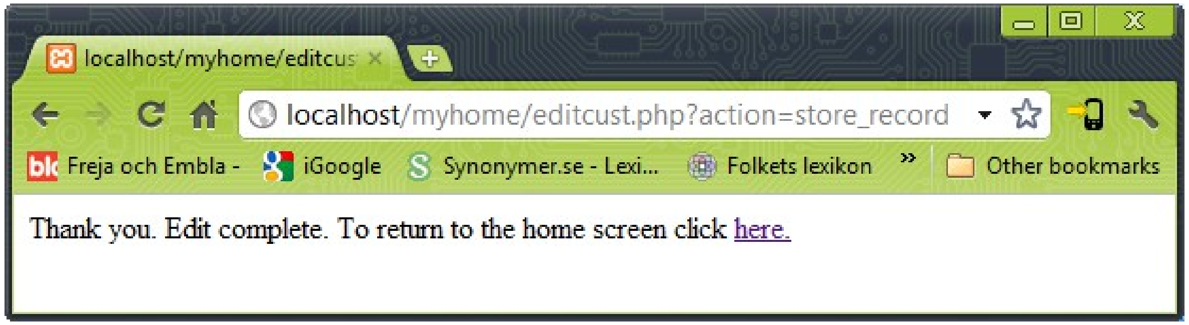
```php
else if ($action == "show_record") {  # Display the customer record form. Populate it with the details of
        # the customer whose id is passed in the id2edit radio button.  Get the contents of the id2edit form variable
        $id2edit = empty($_POST["id2edit"]) ? "" : $_POST["id2edit"];
        if ($id2edit == "") {
                $m = "No customer selected! To return to the home ";
                $m .= "screen click <a href='editcust.php'>here.</a>";
                show_message($m);
        }
        else {
                try{
                        # Now get the customer's details as we'll need them to populate the form
                        $sql = "select * from customers where id = :id2edit";
                        # STH means "Statement Handle"
                        $STH = $DBH->prepare($sql);
                        $STH->execute(array(':id2edit' => $id2edit));
                        # setting the fetch mode, don't need a while loop as there's only 1 row
                        $row = $STH->fetch(PDO::FETCH_ASSOC);
                        printHeader();
                        echo "<form method='POST' action='editcust.php?action=store_record'>";
                        echo "<table>";
                        echo "<tr><td>First Name</td>";
                        echo "<td><input value='{$row["firstname"]}' type='text' ";
                        echo "name='tb_firstname' size='25'></td>";
                        echo "</tr>";
                        echo "<tr><td>Surname</td>";
                        echo "<td><input value='{$row["surname"]}' type='text' ";
                        echo "name='tb_surname' size='25'></td>";
                        echo "</tr>";
                        echo "<tr><td>Title</td>";
                        echo "<td><input value='{$row["title"]}' type='text' ";
                        echo "name='tb_title' size='25'></td>";
                        echo "</tr>";
                        echo "<tr><td>Address</td>";
                        echo "<td><textarea rows='2' name='ta_address' cols='20'>{$row["address"]}</textarea></td>";
                        echo "</tr>";
                        echo "<tr><td>Phone</td>";
                        echo "<td><input value='{$row["phone"]}' type='text' ";
                        echo "name='tb_phone' size='25'></td>";
                        echo "</tr>";
                        echo "<tr><td>Email</td>";
                        echo "<td><input value='{$row["email"]}' type='text' ";
                        echo "name='tb_email' size='25'></td>";
                        echo "</tr>";
                        echo "</table>";
                        echo "<br>";
                        echo "<input type='submit' value='Save' name='button'>";
                        # Pass the id along to the next routine in a hidden field
                        echo "<input type='hidden' name='id2edit' value='{$id2edit}'>";
                        echo "</form>";
                        # close the connection
                        $DBH = null;
                }
                catch (PDOException $e){
                        echo '<b>PDOException: </b>',$e->getMessage();
                        die();
                }
                printFooter();
        }
} # action = show_record
```



First Name	Thomas
Surname	Kvist
Title	Teacher
Address	Gagnef
Phone	12121212
Email	tkv@du.se

Edit a data record 3 - thewebbo_hms

```
else if ($action == "store_record")
{
       # Retrieve the data from the form and update customer record
       $id2edit = $_POST["id2edit"]; # Get the id from the hidden field
       $firstname = $_POST["tb_firstname"];
       $surname = $_POST["tb_surname"];
       $title = $_POST["tb_title"];
       $address = $_POST["ta_address"];
       $phone = $_POST["tb_phone"];
       $email = $_POST["tb_email"];

       # Now we can update the customer's data.
       try{
              $sql = "update customers set firstname=?, surname=?, title=?, address=?,
                     phone=?, email=? where id = $id2edit";
              #echo $sql;
              # STH means "Statement Handle"
              $STH = $DBH->prepare($sql);
              $result = $STH->execute(array($firstname, $surname, $title, $address, $phone, $email));

              $m = "Thank you! Edit of $result row complete. To return to the home ";
              $m .= "screen click <a href='editcust.php'>here.</a>";
              show_message($m);
              # close the connection
              $DBH = null;
       }
       catch (PDOException $e){
              echo '<b>PDOException: </b>',$e->getMessage();
              die();
       }
} # action = store_record

function show_message($s)
{
       printHeader();
       echo $s;
       printFooter();
}
?>
```

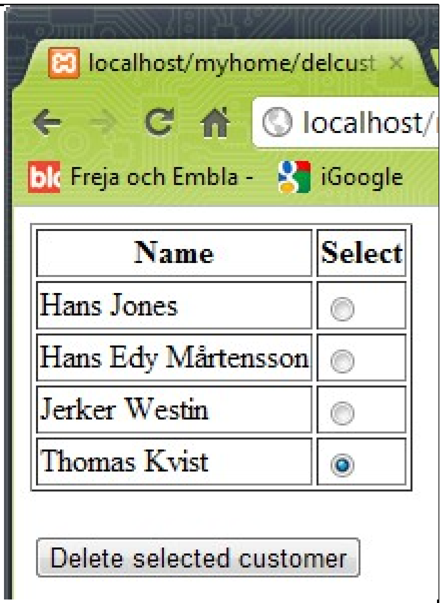


Thank you. Edit complete. To return to the home screen click here.

Delete a data record - thewebbo_hms

- Similar to previous example, be careful, don't delete the whole database!
- A reminder! To retrieve information that was passed as
  - part of the URL → use $_GET['name'];
  - a form field (textbox, textarea, hidden field etc.) → use $_POST['name'];
- Remember that information that you don't explicitly pass either on the URL or in a form field will NOT be present

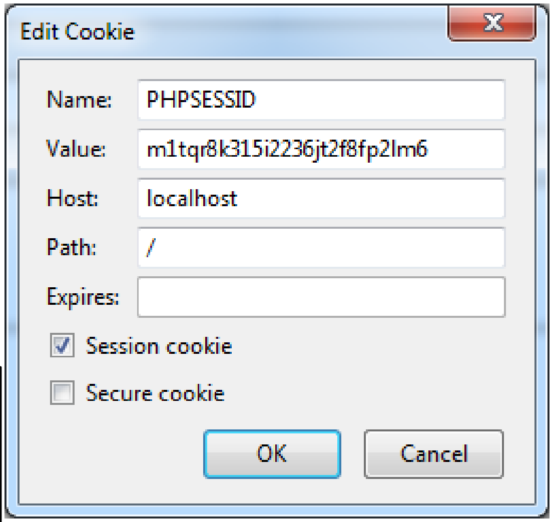
```
else if ($action == "delete_record") {
       # Delete the record
       $id2edit = empty($_POST["id2edit"]) ? "" : $_POST["id2edit"];
       if ($id2edit == "") {
              $m = "No customer selected! To return to the home ";
              $m .= "screen click <a href='delcust.php'>here.</a>";
              show_message($m);
       }
       else {
              try{
                     # Now get the customer's id to delete
                     $STH = $DBH->prepare('delete from customers where id = :id');
                     $STH->bindParam(':id', $id2edit);
                     $STH->execute();
                     $m = "Customer with ID $id2edit deleted. To return to the home ";
                     $m .= "screen click <a href='delcust.php'>here.</a>";
                     show_message($m);
                     # close the connection
                     $DBH = null;
              }
              catch (PDOException $e){
                     echo '<b>PDOException: </b>',$e->getMessage();
                     die();
              }
       }
} # action = delete_record
```



Name	Select
Hans Jones	○
Hans Edy Mårtensson	○
Jerker Westin	○
Thomas Kvist	●

Saving State

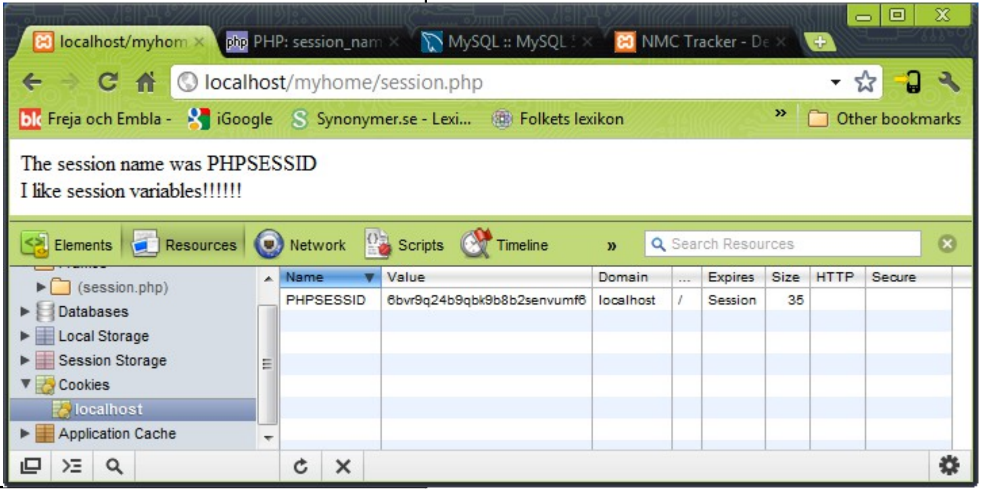
- Sessions are a combination of a server-side cookie and a client-side cookie
- The session cookie reference is passed back and forth between the server and the client
- Server loads the corresponding session data from a unique file on the server



Name:	PHPSESSID
Value:	m1tqr8k315i2236jt2f8fp2lm6
Host:	localhost
Path:	/
Expires:	

[✓] Session cookie
[ ] Secure cookie

```php
<?php
# Initialize session data
# creates a session or resumes the current one based on a session
# identifier passed via a GET or POST request, or passed via a cookie.
session_start();
if(!isset($_SESSION["my_session_var_name1"])) {
     $_SESSION["my_session_var_name1"] = "I like session variables!";
}
else {
      $_SESSION["my_session_var_name1"] .= "!";
}
# Get and/or set the current session name
$sess_name = session_name();
echo "The session name was $sess_name","<br />";
echo $_SESSION["my_session_var_name1"];
?>
<?php
# Ending a session must always be done like this!
    session_start();
    $_SESSION = array();
    session_destroy();
?>
```



localhost/myhome/session.php

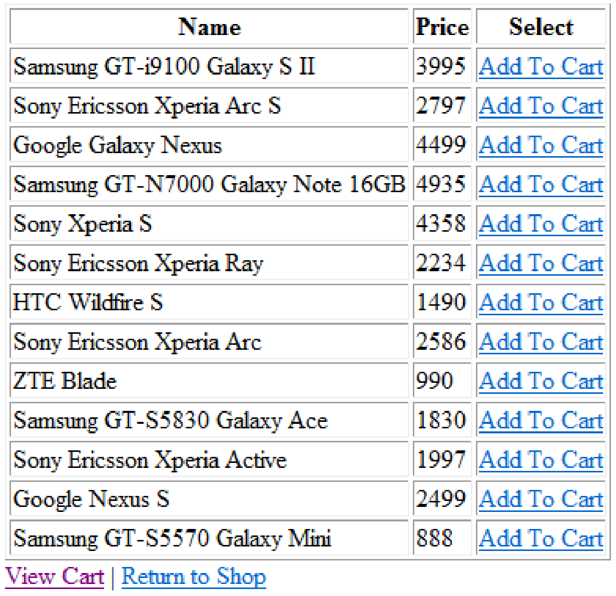
The session name was PHPSESSID
I like session variables!!!!!!

Name	Value	Domain	...	Expires	Size	HTTP	Secure
PHPSESSID	6bvr9q24b9qbk9b8b2senvumf6	localhost	/	Session	35		

Shopping cart - db-shop-content

- Use an unique session variable name on ”utb-mysql.du.se” to avoid collision
- Session behaviour is configured in the [session] part of the php.ini file
  - session.xxx = value
- products.php
  - An ordinary listing of all products
  - SELECT id, name, price FROM products
  - echo "<td><a href=cart.php?action=add&id=",$row["id"],">Add To Cart</a></td>";
- cart.php
  - session_start() - first line!
  - var_dump($_SESSION['cart']);

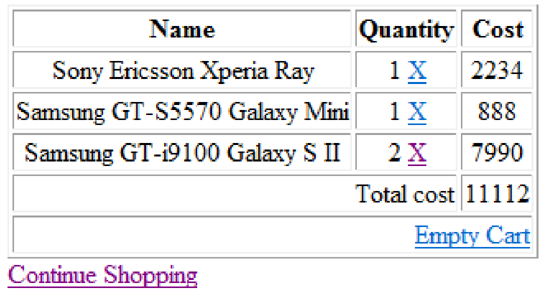
```
products table:
      CREATE TABLE `products` (
            `id` INT NOT NULL AUTO_INCREMENT ,
            `name` VARCHAR( 255 ) NOT NULL ,
            `description` TEXT,
            `price` DOUBLE DEFAULT '0.00' NOT NULL ,
            PRIMARY KEY ( `id` )
      );
```



Name	Price	Select
Samsung GT-i9100 Galaxy S II	3995	Add To Cart
Sony Ericsson Xperia Arc S	2797	Add To Cart
Google Galaxy Nexus	4499	Add To Cart
Samsung GT-N7000 Galaxy Note 16GB	4935	Add To Cart
Sony Xperia S	4358	Add To Cart
Sony Ericsson Xperia Ray	2234	Add To Cart
HTC Wildfire S	1490	Add To Cart
Sony Ericsson Xperia Arc	2586	Add To Cart
ZTE Blade	990	Add To Cart
Samsung GT-S5830 Galaxy Ace	1830	Add To Cart
Sony Ericsson Xperia Active	1997	Add To Cart
Google Nexus S	2499	Add To Cart
Samsung GT-S5570 Galaxy Mini	888	Add To Cart

View Cart | Return to Shop

```
array(3) {
  [6]=>
  int(1)
  [13]=>
  int(1)
  [1]=>
  int(2)
}
```



Name	Quantity	Cost
Sony Ericsson Xperia Ray	1 X	2234
Samsung GT-S5570 Galaxy Mini	1 X	888
Samsung GT-i9100 Galaxy S II	2 X	7990
Total cost		11112
Empty Cart		

Continue Shopping

Shopping cart - db-shop-content

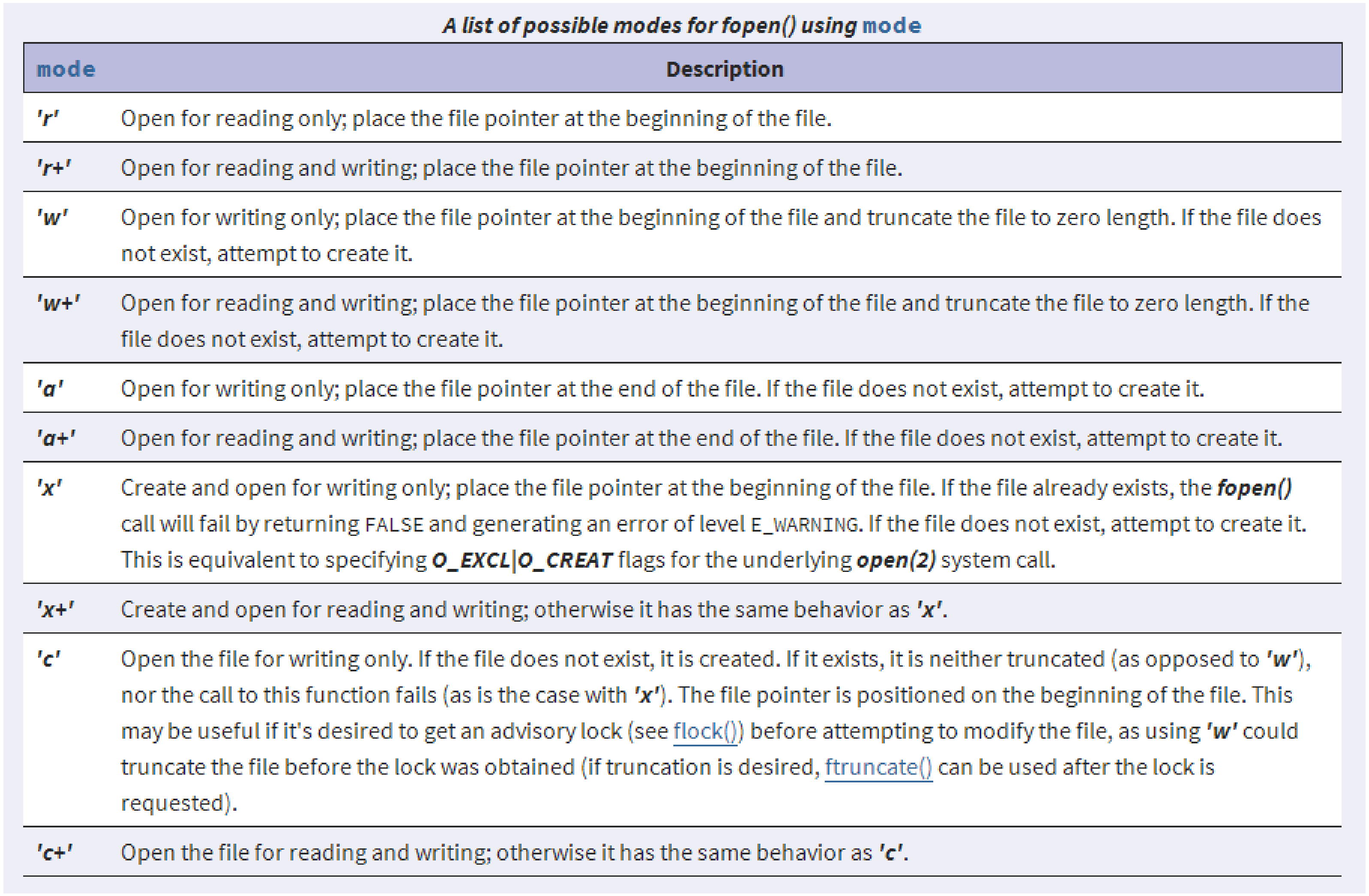
```php
<?php
///........... shortened for brevity  .............
$product_id = empty($_GET['id']) ? "" : $_GET['id']; //the product id from the URL
$action = empty($_GET['action']) ? "" : $_GET['action']; //the action from the URL

if (!isset($_SESSION['cart']))
      $_SESSION['cart'] = array();

//if there is an product_id and that product_id doesn't exist display an error message
if($product_id && !productExists($product_id, $DBH, $products)) {
      die("Error. Product Doesn't Exist");
}

switch($action) { //decide what to do
      case "add":
            if (!isset($_SESSION['cart'][$product_id]))
                  $_SESSION['cart'][$product_id] = 0;
            $_SESSION['cart'][$product_id]++; //add one to the quantity of the product with id $product_id
            break;
      case "remove": //remove one from the quantity of the product with id $product_id
            $_SESSION['cart'][$product_id]--;
            //if the quantity is zero, remove it completely (using the 'unset' function) -
            //otherwise is will show zero, then -1, -2 etc when the user keeps removing items.
            if($_SESSION['cart'][$product_id] == 0)
                  unset($_SESSION['cart'][$product_id]);
            break;
      case "empty":
            unset($_SESSION['cart']); //unset the whole cart, i.e. empty the cart.
            break;
}
///.......... shortened for brevity  .............
?>
```

Files

- Files aren't as flexible as databases, but they do offer the chance to easily and permanently store information across scripts
- Libraries are usually needed for writing binary files (media etc.)
- Path separator? – use ”\\” in Windows or ”/” for all OS:es
- Usually the file function names are taken from Unix or C language

```php
<?php
$file = 'peoples.txt';
$current = file_get_contents($file); // Open the file to get existing content
$current .= "John Smith\n"; // Append another person to the file
// Write the contents back to another file using the FILE_APPEND flag to append the content to the end
// of the file and the LOCK_EX flag to prevent anyone else writing to the file at the same time
file_put_contents("contacts.txt", $current, FILE_APPEND | LOCK_EX);
// file management as rename, copy and delete files
bool rename(string old_name, string new_name [, resource context]);
bool copy(string source, string dest);
bool unlink(string filename, resource context); // delete a file
// read and write to files with fopen(), fread() and fwrite() using the binary flag
$filename = 'contacts.txt';
$handle = fopen($filename, "rb");
$contents = fread($handle, filesize($filename));
fclose($handle);
print $contents;
$handle = fopen('mycontacts.txt', "ab");
$numbytes = fwrite($handle, $contents);
fclose($handle);
print "$numbytes bytes written\n";
?>
```

fopen modes



***A list of possible modes for fopen() using* mode**

mode	Description
'r'	Open for reading only; place the file pointer at the beginning of the file.
'r+'	Open for reading and writing; place the file pointer at the beginning of the file.
'w'	Open for writing only; place the file pointer at the beginning of the file and truncate the file to zero length. If the file does not exist, attempt to create it.
'w+'	Open for reading and writing; place the file pointer at the beginning of the file and truncate the file to zero length. If the file does not exist, attempt to create it.
'a'	Open for writing only; place the file pointer at the end of the file. If the file does not exist, attempt to create it.
'a+'	Open for reading and writing; place the file pointer at the end of the file. If the file does not exist, attempt to create it.
'x'	Create and open for writing only; place the file pointer at the beginning of the file. If the file already exists, the ***fopen()*** call will fail by returning `FALSE` and generating an error of level `E_WARNING`. If the file does not exist, attempt to create it. This is equivalent to specifying ***O_EXCL|O_CREAT*** flags for the underlying ***open(2)*** system call.
'x+'	Create and open for reading and writing; otherwise it has the same behavior as ***'x'***.
'c'	Open the file for writing only. If the file does not exist, it is created. If it exists, it is neither truncated (as opposed to ***'w'***), nor the call to this function fails (as is the case with ***'x'***). The file pointer is positioned on the beginning of the file. This may be useful if it's desired to get an advisory lock (see flock()) before attempting to modify the file, as using ***'w'*** could truncate the file before the lock was obtained (if truncation is desired, ftruncate() can be used after the lock is requested).
'c+'	Open the file for reading and writing; otherwise it has the same behavior as ***'c'***.

Windows offers a text-mode translation flag ('t') which will transparently translate \n to \r\n when working with the file. In contrast, you can also use 'b' to force binary mode, which will not translate your data. To use these flags, specify either 'b' or 't' as the last character of the mode parameter.

Standard PHP Library (SPL) and files

- The Standard PHP Library (SPL) is a collection of interfaces and classes that are meant to solve common problems
  - More or less a wrapper for the standard file system functions
  - Suited for looping over aggregated structures with various iterators
  - http://www.php.net/manual/en/book.spl.php
- SPL provides a number of classes to work with files
  - SplFileInfo - interface to information for an individual file
  - SplFileObject - interface for a file
  - SplTempFileObject - interface for a temporary file – in PHP scripts temp files are often used for storing temporary data
- Resources
  - SPL File Handling
    - http://www.php.net/manual/en/spl.files.php
  - Introduction to Standard PHP Library (SPL)
    - http://www.phpro.org/tutorials/Introduction-to-SPL.html
  - Standard Filesystem Functions
    - http://www.php.net/manual/en/ref.filesystem.php

More file and SPL operations

```php
<?php
// reading and printing out the whole sayhello.txt file in a buffered way
$handle = fopen("sayhello.txt", "r");
while (!feof($handle)) {                    // tests for end-of-file on a file pointer
      $buffer = fgets($handle, 4096); // gets a line from current file pointer
      echo $buffer;
}
fclose($handle);
// reading and printing out every line one by one in file rows.txt
// __construct (string $filename [, string $open_mode = "r" [, bool $use_include_path = false [, resource $cont
$file = new SplFileObject("rows.txt");
foreach($file as $line) {
      echo $line;
}
// reading and printing out a specific line in file rows.txt
$file = new SplFileObject("rows.txt");
$file->seek(3); // seek to specified line
echo $file->current(); // retrieves the current line of the file
$file->fseek(0); // move back to the beginning of the file, same as $file->rewind();
// checking whether a file exists
if (file_exists("fwrite.txt")) {
      $file = new SplFileObject("fwrite.txt", "w");
      $written = $file->fwrite("12345"); // write to a file
      echo "Wrote $written bytes to file";
      // correct close and delete of a SplFileObject – there is no close method or file handle to use
      $file = null;
      unlink($file);
}
// Retrieving a file's status
bool is_readable (string filename);
bool is_writeable (string filename);
bool is_executable (string filename);
bool is_file (string filename);
bool is_dir (string filename);
```

```php
<?php
# a temp file example
$handle = tmpfile();
$numbytes = fwrite($handle, $mystring);
fclose($handle);
print "$numbytes bytes written\n";
?>
```

Write to file via REST

```php
<?php
// call with http://users.du.se/~hjo/lbs/latlng/latlng.php?date=2011/02/14
// &time=17:14:41&wpt=WPT1&lat=60.55&lng=15.55&kmh=15&dir=116&alt=168&userid=13
if(isset($_GET["date"]) && isset($_GET["time"]) && isset($_GET["wpt"])
     && isset($_GET["lat"]) && isset($_GET["lng"]) && isset($_GET["kmh"])
     && isset($_GET["dir"]) && isset($_GET["alt"]) && isset($_GET["userid"]))
{
     $date = $_GET["date"];
     $time = $_GET["time"];
     $wpt = $_GET["wpt"];
     $lat = $_GET["lat"];
     $lng = $_GET["lng"];
     $kmh = $_GET["kmh"];
     $dir = $_GET["dir"];
     $alt = $_GET["alt"];
     $userid = $_GET["userid"];
     $stringData = $date . "," . $time . ","  . $wpt . "," . $lat
          . "," . $lng . "," . $kmh . "," . $dir . "," . $alt;
     echo $stringData;
     // save parameters
     $myFile = "pos" . $userid . ".txt";
     echo " " . $myFile;
     $fh = fopen($myFile, 'a') or die(" can't open file");
     fwrite($fh, $stringData . "\n");
     fclose($fh);
     chmod($myFile,0644); // set permissions rw-r—r-- (owner, group, everyone), only unix
}
else{
     echo "Parameter(s) are missing!";
}
?>
```

Write to file via form and post

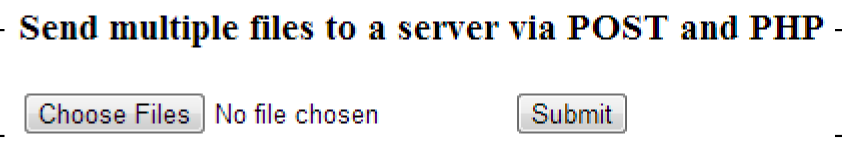
```
<div id="content">
    <?php
    //läser in postat data från formuläret
    if($_POST["regnr"]!="") {
        $bild;//håller sökväg till bild
        //om bild tom ta en defaultbild
        if($_POST["bild"]==""){
            $bild='http://www.bytbil.com/ViewImage.aspx?MediaId=-1&type=prev';
        }
        else {
            $bild=rtrim($_POST['bild']);
        }
        //rtrim=Strip whitespace (or other characters) from the end of a string
        $regnr=rtrim($_POST['regnr']);//läser in postat data från fomulär
        $marke=rtrim($_POST['marke']);
        $modell=rtrim($_POST['modell']);
        $ar=(int)rtrim($_POST['ar']);//typ konverting till heltal,int
        $pris=(int)rtrim($_POST['pris']);
        //struktur över hur bildataraden ska skrivas till filen
        $filrad=$regnr."|".$marke."|".$modell."|".$ar."|".$pris."|".$bild."|\r\n";
        //Filens namn. Den ligger i samma mapp som denna php fil
        $filnamn="bilar.txt";
        //Öppnar en fil
        //a= öppnar för att skriva till slutet av filen, om filen inte finns skapas den.
        $filen=new SplFileObject($filnamn,'a');
        //skriver till filen
        $filen->fwrite($filrad);
    }//end if
    ?>
    <a href="./ShowCarsFromTextFile.php"> Visa alla bilar</a>
</div><!--end content-->
```

Upload file(s) with form and PHP

- Single file?
  - Remove [] in the name attribute and multiple attribute
  - Remove foreach and if in the PHP script as well
- Usually does not work well with large file transfers

```
<form action="uploads.php" enctype="multipart/form-data" method="post">
     <input type="file" name="filesToUpload[]" multiple>
     <input type="submit">
</form>
```

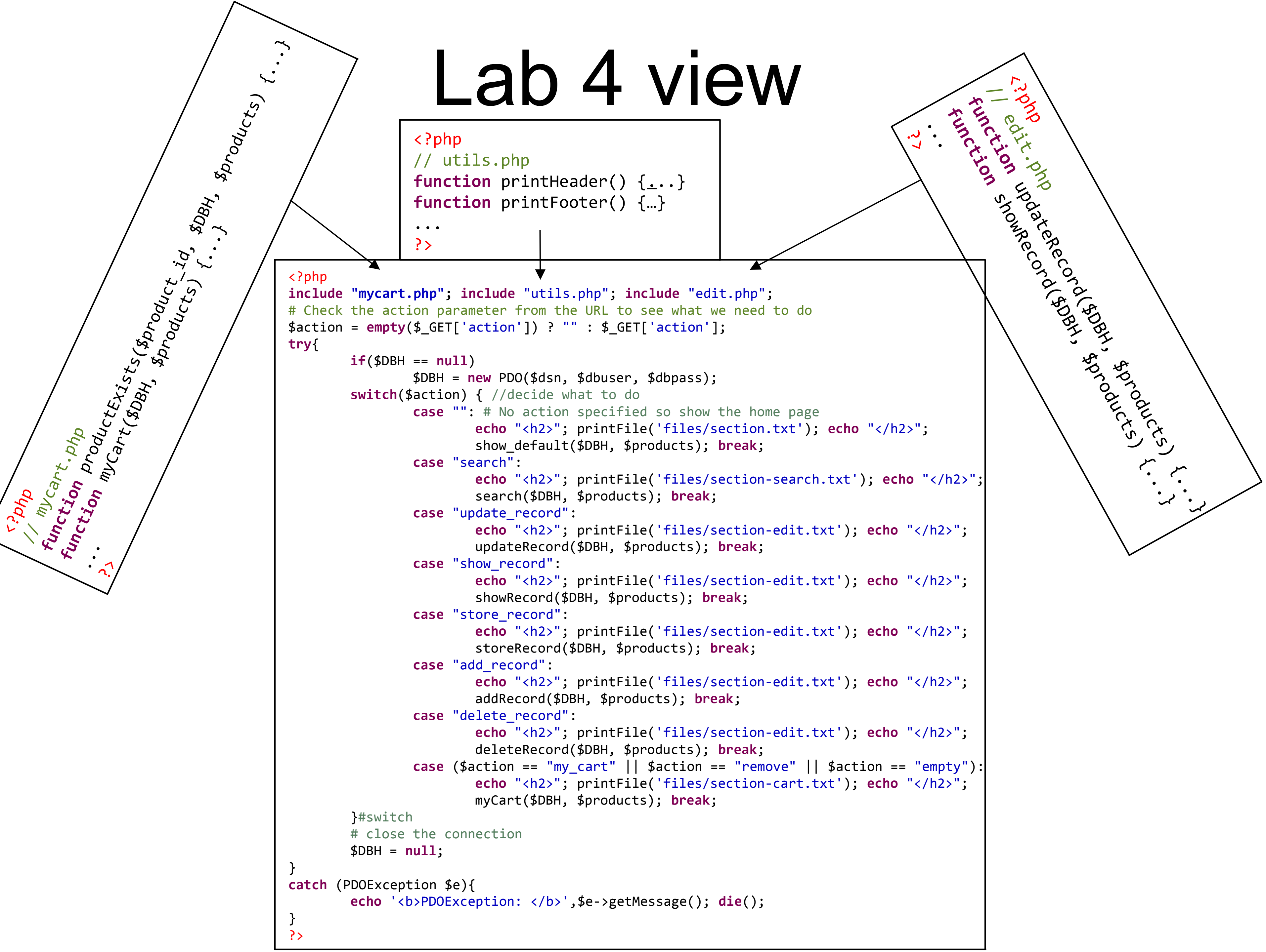
fileupload form code


Send multiple files to a server via POST and PHP

```
<?php
# REMEMBER you have to use the local file system to write files!
$uploads_dir = "D:\\xampp\\htdocs\\myhome";
// $_FILES is an associative array of items uploaded to the current script via the HTTP POST method
foreach ($_FILES["filesToUpload"]["error"] as $key => $error) {
    if ($error == UPLOAD_ERR_OK) {
           $tmp_name = $_FILES["filesToUpload"]["tmp_name"][$key];
           $name = $_FILES["filesToUpload"]["name"][$key];
           // move_uploaded_file() moves an uploaded file to a new location
           if(move_uploaded_file($tmp_name, "$uploads_dir\\$name"))
               echo "The file: ". $name . " has been uploaded <br/>";
           else
               echo "There was an error uploading the file(s), please try again!";
    }
}
?>
```

uploads.php

Lab 4 view



```php
<?php
// mycart.php
function productExists($product_id, $DBH, $products) {...}
function myCart($DBH, $products) {...}
...
?>
```

```php
<?php
// utils.php
function printHeader() {...}
function printFooter() {…}
...
?>
```

```php
<?php
// edit.php
function updateRecord($DBH, $products) {...}
function showRecord($DBH, $products) {...}
...
?>
```

```php
<?php
include "mycart.php"; include "utils.php"; include "edit.php";
# Check the action parameter from the URL to see what we need to do
$action = empty($_GET['action']) ? "" : $_GET['action'];
try{
        if($DBH == null)
                $DBH = new PDO($dsn, $dbuser, $dbpass);
        switch($action) { //decide what to do
                case "": # No action specified so show the home page
                        echo "<h2>"; printFile('files/section.txt'); echo "</h2>";
                        show_default($DBH, $products); break;
                case "search":
                        echo "<h2>"; printFile('files/section-search.txt'); echo "</h2>";
                        search($DBH, $products); break;
                case "update_record":
                        echo "<h2>"; printFile('files/section-edit.txt'); echo "</h2>";
                        updateRecord($DBH, $products); break;
                case "show_record":
                        echo "<h2>"; printFile('files/section-edit.txt'); echo "</h2>";
                        showRecord($DBH, $products); break;
                case "store_record":
                        echo "<h2>"; printFile('files/section-edit.txt'); echo "</h2>";
                        storeRecord($DBH, $products); break;
                case "add_record":
                        echo "<h2>"; printFile('files/section-edit.txt'); echo "</h2>";
                        addRecord($DBH, $products); break;
                case "delete_record":
                        echo "<h2>"; printFile('files/section-edit.txt'); echo "</h2>";
                        deleteRecord($DBH, $products); break;
                case ($action == "my_cart" || $action == "remove" || $action == "empty"):
                        echo "<h2>"; printFile('files/section-cart.txt'); echo "</h2>";
                        myCart($DBH, $products); break;
        }#switch
        # close the connection
        $DBH = null;
}
catch (PDOException $e){
        echo '<b>PDOException: </b>',$e->getMessage(); die();
}
?>
```