Sessions and Cookies management in PHP

Topics

- What are cookies?
- Create and use cookies.
- What are sessions?
- Create and use sessions

Introduction

- HTTP is a stateless protocol.
- A stateless protocol does not require the server to retain information or status about each user for the duration of multiple requests.
- this means that after an exchange is over...
  - a browser requests a resource from a server
  - the web server sends the resource to the browser
- ...the connection will be closed and forgotten
- this has its advantages
  - because there is nothing to be kept track of, it is relatively easy to build web servers that are very efficient
- But also it has drawbacks
  - it makes it hard to follow a user on a website

Why to follow a user's?

- It is often extremely useful to be able to follow a user's activities on a website so that one can for example
  - have a shopping cart
  - maintain the user's identity
  - display information specifically tailored to the individual user
- Increased ability to provide the user a richer experience of using that website
  - a fine example of a site that truly exploits the user's identity is Amazon, which in many ways has set the standard in the application of user identity to provide a rich experience

Login…

- Another aspect is the ability to log into a site with username and password
- this provides some level of security
- One gets the opportunity to have “your own” page
  - with personal things, like your own photos on Facebook
  - a personal configuration of the page
  - a user identity for postings on web boards
- etc.

- Some of the scenarios do not need to know “who” you are
  - “a specific browser on a particular machine” is often enough
  - here we often talk about the temporary nature of information
- In other situations it is useful to know “who” people are
  - one can get a more personalized experience of the site (this helps perhaps also with loyalty)
  - one can log on from different machines and have the same user experience

So what can we do?

- One can add a parameter to the URL and remember it so that all subsequent links on the site contain it
  - but that gives ugly URLs that are difficult to maintain (one always has to rewrite all the URLs in a document) and are vulnerable to trivial hacks
- Since the web server can see, where a request comes from, one could use the user's machine address as an ID
  - but what if it is a shared computer?
  - or if it just looks like one computer due to NAT or a proxy?
- It would be great if a website could save a little bit of data on the user's machine ...

So…

- So, web applications need to track the user's progress from page to page, for example when a web server is required to customize the content of a web page for a user.
- Solutions to address the above mentioned problems are cases that include:
  - the use of HTTP cookies.
  - server side sessions,
  - hidden variables (when the current page contains a form), and
  - URL-rewriting using URI-encoded parameters, e.g., /index.php?session_id=some_unique_session_code.

Solution…

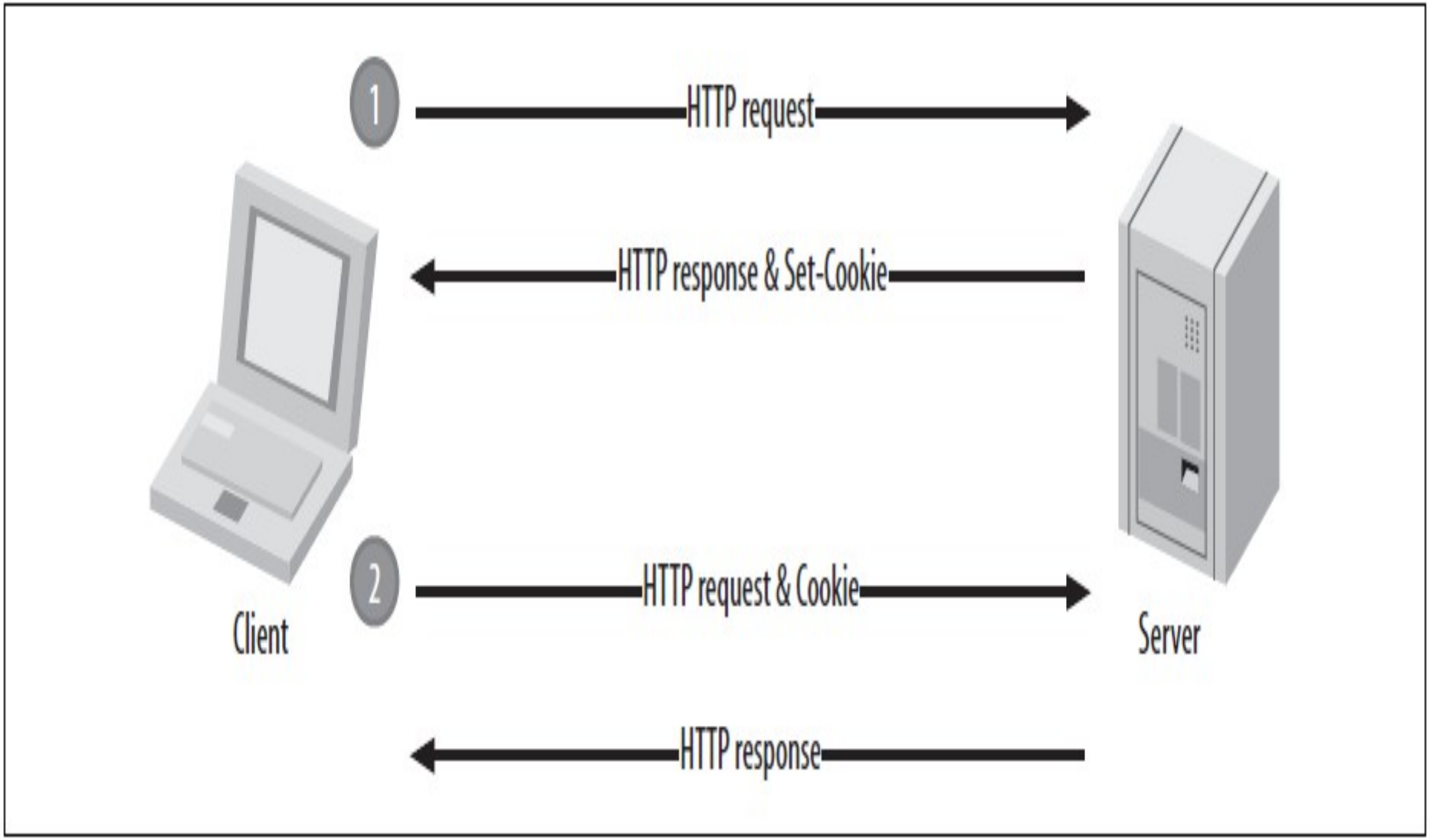
- HTTP remains stateless – there is no fixed connection between web server and browser
- While the stateless nature of HTTP has some important benefits
  - after all, maintaining state requires some overhead
  - it presents a unique challenge to developers who need to create stateful web applications.
- With no way to identify the client, it is impossible to determine
  - whether the user is already logged in,
  - has items in a shopping cart, or needs to register.
- An elegant solution to this problem, originally conceived by Netscape, is a state management mechanism called cookies.

Cookies

- a web server can leave a “cookie” in the browser (i.e. on the user’s computer)
  - it is up to the browser to manage these cookies
  - the cookie gets transmitted to the server in future connections
- A cookie is a small piece of data (typically max. 4 kB, usually far less) that is used by the web server to identify the user
- Cookies may be limited in time with an expiration date
  - else the cookie will be deleted when the browser is closed

Cookies

- are an extension of the HTTP protocol.
- they consist of two HTTP headers:
  - the Set-Cookie response header and
  - the Cookie request header.
- When a client sends a request for a particular URL, the server can opt to include a Set-Cookie header in the response, so as to request for the client to include a corresponding Cookie header in its future requests
- Cookies
  - allow a unique identifier to be included in each request (in a Cookie header),
  - This help to uniquely identify clients and associate their requests together.



A complete cookie exchange that involves two HTTP transactions

Create Cookies in PHP

- Use setcookie() function to create a cookie.

Syntax

setcookie(string *name, string value, int expire, string path,*

string domain, int secure);

Example:

```
<?php
// Setting a cookie
setcookie("username", “Abebe", time()+10*24*60*60);
?>
```

Parameter	Description
name	The name of the cookie.
value	The value of the cookie.
expires	The expiry date in UNIX timestamp format. This implies, After this time cookie will become inaccessible. The default value is 0.
path	Specify the path on the server for which the cookie will be available. If set to '/', the cookie will be available within the entire domain.
domain	Specify the domain for which the cookie is available to e.g www.example.com.
secure	This field, if present, indicates that the cookie should be sent only if a secure HTTPS connection exists.
	Is a Keyword referring that the cookie will not be sent over a plain HTTP connection

```php
if(!empty($_POST["remember"]))
{
    setcookie ("member_login",$name,time()+ (10 * 365 * 24 * 60 * 60));
    setcookie ("member_password",$Password,time()+ (10 * 365 * 24 * 60 * 60));
}
else
{
    if(isset($_COOKIE["member_login"]))
    {
        setcookie ("member_login","");
    }
    if(isset($_COOKIE["member_password"]))
    {
        setcookie ("member_password","");
    }
}
```

Accessing cookie

- The PHP $_COOKIE super global variable is used to retrieve a cookie value.

```
<html>
<head><title>sample on cookie</title></head>
<body>
<?php
if(!isset($_COOKIE["username"]))
{

setcookie("username", "Abebe", time()+10*24*60*60);
}
else{
echo $_COOKIE["username"]; // used to access a
cookie
}?>
</body>
</html>
```

Check if Cookies are Enabled

- count the $_COOKIE array variable

```
<?php
setcookie("username", "Abebe", time()+10*24*60*60);
?>
<html>
<body>

<?php
if(count($_COOKIE) > 0) {
    echo "Cookies are enabled.";
} else {
    echo "Cookies are disabled.";
}
?>
</body>
</html>
```

Removing Cookies

Cookies can be deleted by calling the ***setcookie()*** function with the cookie name and any value (such as an empty string) with expiration date set in the past,

```
<?php
// Deleting a cookie
setcookie("username", "", time()-3600);
?>
```

Scope of cookies

- Cookies can only be read from the site from which they were set
- this helps to ensure that one can not steal cookies (and thus identities) through hostile websites

Cookies and their use

- Shopping cart
  - when the front page appears, set a new (empty) cookie
  - items are added by updating the cookie
  - *alternatively, one can store goods in the server's database and just store an ID in the cookie that points to your basket*
- Login
  - user inputs name and password into a form
  - *after the combination has been verified, it sends a cookie to the user that identifies the user to the system*
  - next time the user visits the page, the web server checks if there is a cookie, and if so the user is identified

Criticism of cookies

- One can not be completely anonymous on the net
  - most ads/banners come from relatively few advertisers.
  - these may, with the help of cookies, follow a browser/ computer combination on all the sites they advertise on.
- there have been examples of security vulnerabilities in browsers, so that adversaries can get access to cookies
  - which they can use to gain access to sites with a faked identity

Sessions

- A combination of cookies and data stored on the server (automatically by PHP)
  - saves a cookie containing an ID on the user's computer that points to a session on the server
- A session is a global PHP array ($_SESSION)
- A session is designed as an easy way to store data – for a short period
  - a session's lifetime in PHP is only 24 minutes by default

Starting PHP session

- Sessions must be started at the top of the page before it is used

  session_start()

- Session_start() function creates a new session and generate a unique session ID for the user.
  - it first checks for an existing session ID. If it finds one, i.e. if the session is already started, it sets up the session variables and if doesn't, it starts a new session by creating a new session ID.

Storing and Accessing Session Data

- session data can be stored as key-value pairs in the $_SESSION[] super global array.
  - The stored data can be accessed during lifetime of a

```
<?php
// Starting session
session_start();

// Storing session data
$_SESSION["firstname"] =
“Abebe";
$_SESSION["lastname"] = “Lemlem";
?>
```

Accessing session data

- We can access the session data we set on our previous example from any other page on the same web domain
- Simply recreate the session by calling session_start() and then pass the corresponding key to the $_SESSION associative array.

```
<?php
session_start();
echo $_SESSION["Name"];
?>
```

Destroying PHP session

- all global session variables can be removed by destroing the session using
  - session_unset(“sessionId”) //remove all session variables
  - session_destroy():// destroy the session

```
<?php
session_start();
// Removing session data
if(isset($_SESSION[“username
"])){
session_unset($_SESSION[“us
ername"]);
}
?>
```

```
<?php
session_start();
// Destroying
session
session_destroy()
;
```

Session for login

- Sessions are tied to an individual user and a corresponding browser
- therefore, they are well suited to handle logins
- Useful command in this context: header()
- **header()** *is used to send HTTP headers to the browser*
- *as other header information (such as cookies and sessions), such a command should precede any HTML in a PHP file*
- *a particularly interesting header in this context is* **Location:***, which redirects the browser to another URL*

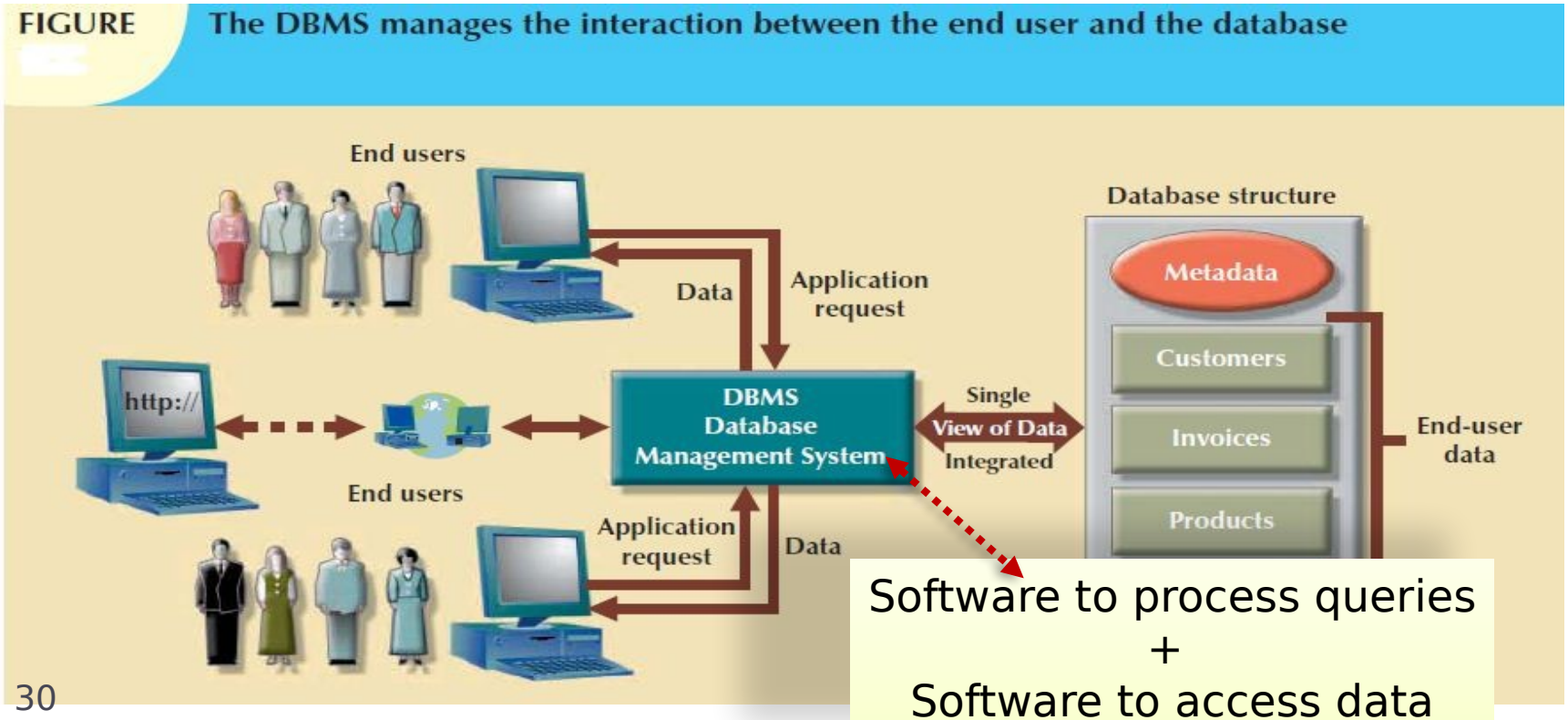
```php
<?php
session_start();
echo session_id() . '<br>';
if(isset($_SESSION['counter'])){
    $_SESSION['counter']++;
}else{
    $_SESSION['counter']=1;
    }
?>

<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
        content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
<h1>Page visited:<?php echo $_SESSION['counter'] ?> Times</h1>
</body>
</html>
```

Data Base manipulation using PHP

Review on Database terms

- **Database:** is a collection of related data
- **Database Management System (DBMS)**: A software package/ system to facilitate the creation and maintenance of a computerized database.



Databases and PHP

- Currently PHP can communicate with different DBMS’s

Connection with database can be achieved either through

- Vendor specific extensions such as MySQL Drivers and Plugins or
- DAB’s -database abstraction layers such as, PDO, ODBC, dbx, DBA

Introduction to MySQL

- MySQL is a RDBMS
- Is an open-source software relational database management system
- Use standard form of SQL data language

- Assignemt – 1 : submission deadline Jan 20, 2021
  - Select one NoSQL database of your preference and prepare a hand written note with not more than 2 pages on the features of the DB and compare it to that of relational database
  - Consider the selected NoSQL database and write a working PHP code to INSERT, UPDATE, DELETE and SELECT(retrive a data)

Basic queries

- CREATE: create databases and tables
  - *CREATE TABLE tableName(columnName Datatype constraint, ...);*
- SELECT: select table rows based on certain conditions
  - *SELECT columnName, columnName... FROM tableName WHRER condition;*
- DELETE: delete one or more rows of a table
  - *DELETE FROM tableName WHERE condition;*
- INSERT: insert a new row in a table
  - *INSERT INTO tableName(column list) VALUES (column values);*
- UPDATE: update rows in a table
  - *UPDATE tableName SET columnName = NewValues;*
- ALTER: alter the structure of a table
  - *ALTER TABLE tableName ADD columnBame dataType;*
  - *ALTER TABLE tableName DROP columnName;*
  - *ALTER TABLE tableName MODIFY columnName Datatype [NULL value] [DEFAULT value];*
  - *ALTER TABLE tableName CHANGE old_columnName new_columnName datatype;*
  - *ALTER TABLE tableName RENAME TO new_tableName;*

Administrative MySQL Command:

- **USE *Databasename***: This will be used to select a particular database in MySQL work area.
- **SHOW DATABASES:** Lists the databases that are accessible by the MySQL DBMS.
- **SHOW TABLES:** Shows the tables in the database once a database has been selected with the use command.
- **SHOW COLUMNS FROM *tablename:*** Shows the attributes, types of attributes, key information, whether NULL is permitted, defaults, and other information for a table.
- **SHOW INDEX FROM *tablename*:** Presents the details of all indexes on the table, including the PRIMARY KEY.
- **SHOW TABLE STATUS LIKE *tablename*\G:** Reports details of the MySQL DBMS performance and statistics.
- **Create user:** CREATE USER 'newuser'@'localhost' IDENTIFIED BY 'password';

Grant privilege: GRANT type_of_permission ON database_name.table_name TO 'username'@'localhost';

Reload privileges: FLUSH PRIVILEGES;

revoke permission: REVOKE type_of_permission ON database_name.table_name FROM 'username'@'localhost';

View permission: SHOW GRANTS FOR 'username'@'localhost';

PHP and MySQL common functions

- Connecting to a Database
- Making a query
- Using results of a query
- Freeing resources
- closing the connection

Connecting to a Database

- Opens a new connection MySQL server
- Can be done through mysqli_connect(procedural style), ***mysqli*** (OOP) or by data abstraction layer (***PDO***)

```php
<?php
function connect_db($dbname)
{
define("HOST", "localhost");
define("USER", "root");
define("PWD", "passwd");
define("DB_NAME", $dbname);

$mysqli = new mysqli(HOST, USER, PWD, DB_NAME);

if($mysqli->connect_errno){
    echo "connection failed: " . $mysqli->connect_error;
    exit();
}
return $mysqli;
}
?>
```

```php
public PDO $pdo;

public function __construct()
{
    $this->pdo = new PDO('mysql:server=localhost;dbname=notes','root', 'passwd');
    $this->pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
}
```

Making a query (***1. SELECT***)

```php
function view(){
$con = connect_db('notes');
$query = "SELECT * FROM users";
$result = $con->query($query);
return $result;
```

```php
public function getNotes(){
    $statment = $this->pdo->prepare("SELECT * FROM notes ORDER BY create_date");
    $statment->execute();
    return $statment->fetchAll(PDO::FETCH_ASSOC);
}
```

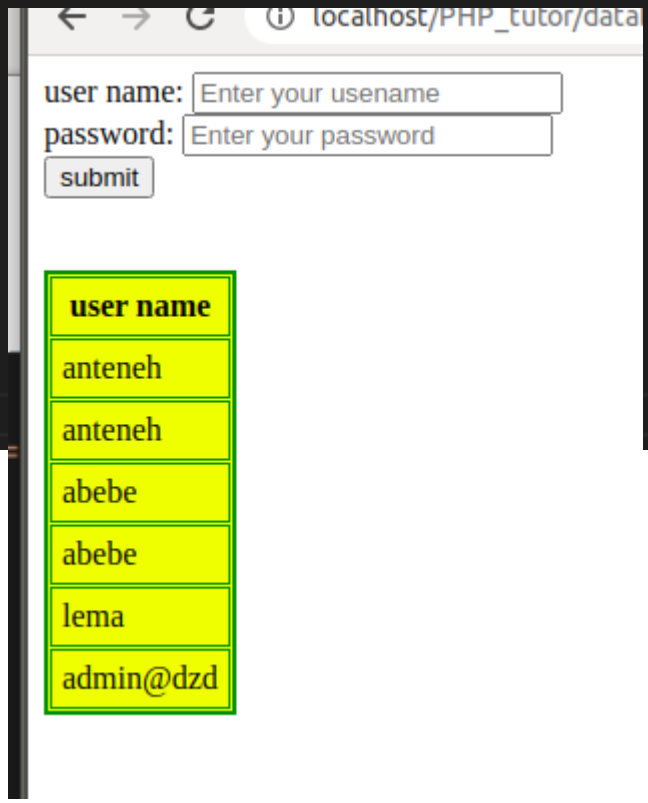
Using the query result : example

```php
<?php
echo '<table border="2" cellpadding="5" cellspacing="1" bordercolor ="green" bgcolor="yellow">';
echo '<tr><th> user name</th></tr>';

include('select_mysqli.php');

$result = view();

while($r= $result->fetch_assoc())
{
    echo '<tr><td>'. $r['username'] . '</td></tr>';
}
echo '</table>';
?>
```



Making a query : insert

```php
$con = connect_db('notes');
$uname=$con->real_escape_string($_POST['username']);
$passwd = $con->real_escape_string($_POST['password']);

$query = "INSERT INTO users(username, password) VALUES('$uname', '$passwd')";
$result = $con->query($query);
if(!$result)
{
    $err =1;
}

$con->close();

if(isset($err))
{
    header("location: index.php?err=1");
}
else
{
    header("location: index.php");
}
```

Making a query : insert, update - PDO

```php
public function addNote($note)  {

    $statment = $this->pdo->prepare("INSERT INTO notes(title, description, create_date) VALUES(:title, :description, :date)");
    $statment->bindValue('title', $note['title']);
    $statment->bindValue('description', $note['description']);
    $statment->bindValue('date', date('Y-m-d H:i:s'));
    return $statment->execute();
}
```

```php
public function updateNote($id, $note){

    $statment= $this->pdo->prepare("UPDATE notes SET title = :title, description = :description WHERE id= :id");
    $statment->bindValue('id', $id);
    $statment->bindValue('title', $note['title']);
    $statment->bindValue('description', $note['description']);
    return $statment->execute();
}

public function deleteNote($id){
    $statment= $this->pdo->prepare("DELETE FROM notes WHERE id= :id");
    $statment->bindValue('id', $id);
    return $statment->execute();


}
```

Freeing query resources and closing connection

- free memory associated with the given resource
- called result (after a select query).
- Not necessary except for large result sets
- Done automatically when script exits.

```
$result->close();
```

closing the connection : close the database connection associated with the given database link

- `$con->close($con)`

Other functions

$con->real_escape_string($toEscapeString);

- This function is used to create a legal SQL string that you can use in an SQL statement.
- The given string is encoded to an escaped SQL string, taking into account the current character set of the connection.
- parameters
  - $toEscapeString: the string to be escaped.
    - Characters encoded are NUL (ASCII 0), \n, \r, \, ', ", and Control-Z.

Error handling

mysqli_connect_errno()

- Returns the last error code number from the last call to mysqli_connect
- An error code value for the last call to mysqli_connect, if it failed.
- zero means no error occurred

mysqli_connect_error

- Returns a string description of the last connect error
- NULL is returned if no error occurred.

Reference

- For more detail review the PHP manual https://www.php.net/manual/en/refs.database.php
-