PHP

EXERCISES

home / exercises / php

#exercises #php #web

PHP EXERCISES

1. FIRST PHP SCRIPT

- Check if your working area (samba or Z drive) contains a folder called *public_html*. If it does not, create it.
  - **Tip**: Click here if you need **help** accessing your working area.
- Inside that folder, create a file called **exercise1.php**.
- Inside that file write the following code:

```
<?php echo "<h1>Hello World!</h1>"?>
```

- Verify if your script works by navigating to http://gnomo.fe.up.pt/~upxxxxxx/exercise1.php (replace *upxxxxxx* with your own student code).
- Some alternative ways of getting the same result:

```
<h1>
<?php echo "Hello World!" ?>
</h1>
```

Or even:

```
<h1><?= "Hello World!" ?></h1>
```

2. HTTP PARAMETERS

- Create a script called **sum2.php** that receives two numbers (**num1** and **num2**) as arguments and prints the sum of those two numbers. To access the arguments you should use the special **$_GET** array.
- Try your script by going to http://gnomo.fe.up.pt/~upxxxxxx/sum2.php?num1=2&num2=5 (replace *upxxxxxx* with your own student code).
- Modify the script so that the output is valid HTML.
- Create a new HTML file, called **form2.html**, that contains a form asking for two numbers. Make that form send the numbers to the script you created previously.
- Modify the PHP file so that it contains a link back to the form.

3. SQLITE DATABASE CREATION

- Establish a SSH connection to *gnomo.fe.up.pt*. If you are using *Windows* you can use Putty. If you are using *Linux* or a *Mac*, just execute the following command in the terminal (replace *upxxxxxx* with your own student code):

```
ssh upxxxxxx@gnomo.fe.up.pt
```

- Enter your *public_html* folder:

```
cd public_html
```

- Create a new folder called **exercise3**:

```
mkdir exercise3
```

- Enter the new folder:

```
cd exercise3
```

- Copy the file news.sql into this folder. You can use samba to do this, or download the file directly using the following command:

```
wget
http://paginas.fe.up.pt/~arestivo/page/files/exercises/php/news.sql
```

- This file contains a SQL script that creates a new database. Open the file and see if you can understand what it does.
- Create a new database inside the *exercise3* folder using the command :

```
sqlite3 -init news.sql news.db
```

- After this last command, the *SQLite* interface will become active. To verify if the database has been created correctly, try some SQL commands:

```
select * from news where id = 4;
select * from comments where news_id = 4;
```

- To exit from the *SQLite* interface just type:

```
.exit
```

- Verify if a new file called **news.db** has really been created using the command:

```
ls
```

- To go back to the *SQLite* interface and interact with the new database just type:

```
sqlite3 news.db
```

4. LISTING DATA FROM SQLITE

- Create a new PHP file, called *list_news.php*, in the same folder where you created your SQLite database.
- Open a connection to the database using the following PHP code:

```php
<?php
  $db = new PDO('sqlite:news.db');
?>
```

- The **$db** variable now represents your connection to the database:
- Execute a query returning all news in the database using the following code:

```php
$stmt = $db->prepare('SELECT * FROM news');
$stmt->execute();
$articles = $stmt->fetchAll();
```

- The variable **$articles** is now an associative array containing all the news in the database. You can iterate over this array using:

```php
foreach( $articles as $article) {
  echo '<h1>' . $article['title'] . '</h1>';
  echo '<p>' . $article['introduction'] . '</p>';
}
```

- Verify the result in http://gnomo.fe.up.pt/~upxxxxxx/exercise3/list_news.php (replace *upxxxxxx* with your own student code).

5. COMPLETE PAGE

- Using your recently acquired knowledge in PHP, change the **list_news.php** script so that it creates a page like the one we used in the first CSS exercise. But this time, news should be retrieved from the database instead of being hard coded.
- You can use the following SQL to get all the information about each article:

```
SELECT news.*, users.*, COUNT(comments.id) AS comments
FROM news JOIN
     users USING (username) LEFT JOIN
     comments ON comments.news_id = news.id
GROUP BY news.id, users.username
ORDER BY published DESC
```

- Do not forget to verify if the page returns valid HTML code.
- Also create a new page, called **news_item.php** that receives a parameter **id** containing the *id* of a news item in the database. This page should be able to present that news item and its comments. Use the following code to get the item information:

```
  $stmt = $db->prepare('SELECT * FROM news JOIN users USING (username)
WHERE id = :id');
  $stmt->bindParam(':id', $_GET['id'], PDO::PARAM_INT);
  $stmt->execute();
  $article = $stmt->fetch();
```

or:

```
  $stmt = $db->prepare('SELECT * FROM news JOIN users USING (username)
WHERE id = ?');
  $stmt->execute(array($_GET['id']));
  $article = $stmt->fetch();
```

- And this to get the comments:

```
  $stmt = $db->prepare('SELECT * FROM comments JOIN users USING
(username) WHERE news_id = ?');
  $stmt->execute(array($_GET['id']));
  $comments = $stmt->fetchAll();
```

- Create links from the **list_news.php** page to the **news_item.php** page in the correct places.
- Validate both pages.

6. DATA LAYER SEPARATION

- Create a new folder called **database** and inside that folder a new file called **connection.php**. Copy the code that you used to initialize the database into the new file.
- Replace the initialization code in all pages with an **include_once** instruction that includes the newly created file.
- Inside the same folder, create a file called **news.php**. This file will be responsible for all accesses to the **news** table.
- Inside this file create a function called **getAllNews** that queries the database and returns an array with all the news in the database.
- Replace the code in your pages where all the news are retrieved by from the database with calls to the new function. The initial PHP code in your **list_news.php** should now look like this:

```
<?php
  include_once('database/connection.php');
  include_once('database/news.php');
  $articles = getAllNews();
?>
```

- Do the same changes in all your pages. Replace all database connection code with includes of the **connection.php** page and all queries by calls to functions defined in the **news.php** file. Functions related to the comments table should be created in a new file called **comments.php** in the **database** folder.

7. PRESENTATION LAYER SEPARATION

- All visual pages start and end with the exact same HTML code. Copy that code into two files called **header.php** and **footer.php** inside a new folder called **templates/common**.
- Replace that code in **all** files by *includes* of those new files.
- For the HTML code specific to each page create new files in the templates folder called, for example, **news/list_news.php**, **comments/list_comments.php** and **news/view_news.php**.
- Replace that code in **all** files by including those new files.
- The **list_news.php** page should now look like this:

```
<?php
  include_once('database/connection.php'); // connects to the database
  include_once('database/news.php');       // loads the functions
responsible
                                           // for the news table
  $articles = getAllNews();                // gets all news from the
database

  include('templates/common/header.php');  // prints the initial part
of the HTML document
  include('templates/news/list_news.php'); // prints the list of news
in HTML
  include('templates/common/footer.php');  // prints the initial part
of the HTML document
?>
```

- From this point forward all pages should be written using this format.

8. EDITING DATA

- Create a new page called **edit_news.php** that contains a form with 3 fields: **title**, **introduction** and **fulltext**. Use an input with type text for the first one and textareas for the other two. Also add a submit button.
- This page should receive a parameter called **id** containing the id of the news to be edited.
- Add a *hidden* input field containing this **id**.
- Retrieve the news item to be edited from the database and fill the fields with its values.
- Add a link in the **news_item.php** file that points to this new page.
- The action of this new form should point to a new page called **action_edit_news.php**.
- This new page should receive the four values (id, title, introduction and fulltext) and update the news table with them. Don't forget to separate this code into the **database/news.php** file.
- This new page should not print any HTML. Instead, it should redirect the user back to the **news_item.php** page sending the correct **id** as a parameter.

9. AUTHENTICATION

- Create a new table in the database with two text columns, username and password, called users. Add some values. Don't forget that passwords should be saved using, for example, SHA1. You can use this website for now to calculate the values to store in the database: http://www.sha1-online.com/
- Create a new page called **login.php** asking for a username and a password.
- This new form should send the data (username and password) to a **action_login.php** page.

- In this new page start by initiating the session, verifying if the username and password are corrects and, if they are, storing the username in the session. In the end redirect back to the page you came from. Something like this:

```php
<?php
  session_start();                              // starts the session
  include_once('database/connection.php'); // connects to the database
  include_once('database/users.php');      // loads the functions
responsible for the users table

  if (userExists($_POST['username'], $_POST['password']))  // test if
user exists
    $_SESSION['username'] = $_POST['username'];            // store the
username

  header('Location: ' . $_SERVER['HTTP_REFERER']);
?>
```

- Add a **session_start** in all pages so you can access the current user easily in $_SESSION['username'].
- Change the header template so that if the $_SESSION array contains a valid user it won't show links to the login and logout pages but to a logout link pointing to a **action_logout.php** page.
- Create this page so that it destroys the session and redirects back to the previous page.
- Change the **news_item.php** page so that the link to edit news only appears if a user is logged in.
- Change the **edit_news.php** and **action_edit_news.php** pages so that if a user is not logged in it will redirect back to the main page.

10. INSERT AND DELETE

- Create pages to insert and delete news.

SOLUTIONS

Solutions for these PHP exercises.