PHP

Sumanth Ratna

May 13, 2019

1 Introduction

- **PHP** Hypertext Preprocessor
- backend/server-side
- easy to use when communicating with databases
- **Why?** there are tools like Node.js which work in backend, but it comes down to personal preference (most backend tools have similar speed)

2 Using PHP

1. navigate to https://director.tjhsst.edu/
2. create a new site
   - if you've created a site before, click "Create Site" at the top-right corner
   - if you haven't created a Director site before, there should be 4 big buttons
3. fill out the fields accordingly, and under the category dropdown click "PHP"

3 Useful Things to do with PHP

Here are two pretty cool and practical things to do with PHP.

3.1 Creating a login system

1. We want it to be secure, so we need a database. Create a mySQL database.
2. Create `index.php`, `login.php`, `logout.php`, `secured.php`, `session.php`, and `config.php`
   - `login.php` will show the login page
   - `logout.php` will log the user out

- `session.php` manages the user session
- `config.php` is for the database
- `secured.php` has the actual (secure) content

3. Create 2 HTML inputs for the username and password in `login.php` and a submit button (that sends a POST request to itself). I'd recommend an HTML form to deal with this.

4. We can use PHP sessions to deal with keeping track of the user's login state. In `session.php` add the following:

```
<?php
require_once 'config.php';
session_start();

$check = $_SESSION['user'];

$ses_sql = mysqli_query($db, "select username from users where
   username = '$check' ");

$row = mysqli_fetch_array($ses_sql, MYSQLI_ASSOC);

$login_session = $row['username'];
if (!isset($_SESSION['user'])) {
    header("location:login.php");
    die();
}
?>
```

5. We need to set up our database. In the mySQL command line, enter the following:
`USE <DATABASE NAME>`

`CREATE TABLE users(id INT NOT NULL AUTO_INCREMENT, username VARCHAR(255) NOT NULL, password VARCHAR(255) NOT NULL, PRIMARY KEY(id))`

`INSERT INTO users (username, password) VALUES ('admin', SHA2('password', 512))`

6. We have the database set up, but we need to save URL, password, and all of the essential information. There should be a folder called `private`, next to `public`. In `private`, create `secrets.php` with the following contents:

```
<?php
$database_url = <THE URL>;
$database_name = <THE NAME>;
$database_password = <THE PASSWORD>;
?>
```

7. In `config.php` add the following code:

```
<?php
require_once '../private/secrets.php';
$db = mysqli_connect($database_url, $database_name,
   $database_password, $database_name);
?>
```

8. When the user attempts to log in, we need to validate their credentials. At the top of `login.php` add the following code:

```
<?php
require_once 'config.php';
session_start();
if (isset($_SESSION['user'])) {
    header("location: secured.php");
    die();
}
if ($_SERVER["REQUEST_METHOD"]=="POST") {
  if (isset($_POST['login'])) {
    $username = mysqli_real_escape_string($db, $_POST['login'
       ][0]);
    $password = mysqli_real_escape_string($db, $_POST['login'
       ][1]);
    $sql = "SELECT id FROM users WHERE username = '$username'
       and password = SHA2('$password', 512)";
    $result = mysqli_query($db, $sql);
    if (!$result) {
        printf("Error: %s\n", mysqli_error($db));
        die();
    }
    $row = mysqli_fetch_array($result, MYSQLI_ASSOC);
    $active = $row['active'];
    $count = mysqli_num_rows($result);
    if($count == 1) {
        $_SESSION['user'] = $username;
        header("location: secured.php");
    }
    else {
        $message = "Invalid credentials";
    }
  }
}
?>
```

Check out the variable named `$sql` and notice how an easy SQL injection attack won't work with it.

Under your HTML inputs and button, add `<?php echo $message; ?>` so you can output error messages to the webpage.

9. When the user logs out, we need to kill their session. In `logout.php` enter

```
<?php
  session_start();

  if(session_destroy()) {
      header("Location: login.php");
  }
?>
```

10. In `index.php` use PHP to redirect to `login.php`.

11. Add some content to `secured.php`. To log the user out, redirect them to `logout.php`. Whenever you need to access the `$_SESSION` global variable, at the very top of the file, add `<?php session_start(); ?>`.

The login system is completed (and secure).

3.2 Contact Form

Let's say you only want authorized users to contact you through your website.

- Add `$sendgrid_key` to `secrets.php`.
- In `secured.php` add the following code to the top:

```
<?php
require_once '../private/secrets.php';
$message = '';
if ($_SERVER['REQUEST_METHOD']=="POST" and isset($_POST['contact
    ''])) {
  require_once 'vendor/autoload.php';
  $data = $_POST['contact'];
  $email = new \SendGrid\Mail\Mail();
  $email->setFrom('sumanthratna@gmail.com');
  $email->setSubject('12 reasons why php is cool');
  $email->addTo("sratna@sumanthratna.gq", "Sumanth Ratna");
  $email->addContent("text/html", nl2br($data[0]).'<br><br><br><
     br>--------------------<br>'.'- anonymous');
  $sendgrid = new \SendGrid($sendgrid_key);
  try {
      $response = $sendgrid->send($email);
      $message = 'Thanks for your message!';
  } catch (Exception $e) {
      $message = 'error sending message:\n\n'.$e->getMessage().'
         \n\nHere\'s what you wrote:\n'.nl2br($data[0]);
  }
}
?>
```

In order to make this work, you'll need to run this inside the `public` folder in your terminal: `composer require sendgrid/sendgrid`.

Obviously, this contact form could be improved since there's no real "from" information, but it's not difficult to implement.

If you want to verify emails before sending (so you don't waste API calls to SendGrid), there's a PHP API for NeverBounce.

- In `secured.php` you'll also need an HTML textarea to get the user's message.