jesuitas
Logroño

SERVER-SIDE WEB PROGRAMMING UNIT5: STORING INFORMATION WITH DATABASES

CFGS - DAW 2

Index

1. Managing Authentication

1. In your bookstore DB, follow these steps:
   A. Create this new table called *users*:

#	Nombre	Tipo
1	id	int(11)
2	name	varchar(255)
3	email	varchar(255)
4	password	varchar(255)
5	updated_at	timestamp
6	created_at	timestamp

   B. Make *id* from *files* primary key from that table.

1. Managing Authentication

2. Now that you have the new table created, you will have to add a new controller class called *UsersController* , the same way you did with Books and Files:
   - Inside we will have three methods: *register*, *login and logout*. Just prepare the structure.

1. Managing Authentication

3. Apart from that, you will have to add a new model class called *Users*, the same way you did with Books and Files:
   - Inside we will have two methods: *register* and *login*. Just prepare the structure.

Can you imagine why, I did not tell you to create a logout method in the model?

1. Managing Authentication

4. Let´s start by completing the register method. Follow this steps:
    1. Inside of the method write a condition that checks in the REQUEST_METHOD is a POST (and leave what is inside empty for now).
    2. In case it is not or there is an error (after the condition), send the user to an *register.html.php* view.
    3. That view, will include a form with the corresponding fields. Create it inside of the corresponding folder.
    4. Also, modify your layout “Register” link in order it to call to …/users/register.

Test that the link in the layout is working and that it loads the register.html.php view.

1. Managing Authentication

5. Now, let´s control in the register method from the controller what happens if a POST request reaches:
    1. Make a static call to the Users/register model method.
    2. Through that variable called to the register method from the model.

1. Managing Authentication

3. On your own, complete the register method from the model, following the same structure of Books/add method:
    1. Sanitize the input
    2. Encrypt the incoming password using md5() method.
    3. In case any of the fields are empty print an error.
    4. If there is no error: save the new user and redirect the user to the …/books/login page.

Test the …/books/register POST action and see if you are saving the information in the DB. Check also if the password is encrypted.

1. Managing Authentication

6. Now let´s complete the login action:
   1. Inside of the method write a condition that checks in the REQUEST_METHOD is a POST (and leave what is inside empty for now).
   2. In case it is not or there is an error (after the condition), send the user to *login.html.php* view.
   3. That view, will include a form with these fields: email and password. Create it inside of the corresponding folder.
   4. Also, modify your layout "Login" link in order it to call to …/users/login.

Test that the link in the layout is working and that it loads the login.html.php view.

1. Managing Authentication

7. Now, let´s control in the login method from the controller what happens if a POST request reaches:
    1. Create a new instance of the Users model and save it in a variable.
    2. Through that variable called to the login method from the model.

1. Managing Authentication

3. On your own, complete the login method from the model, following the same structure of Books/add method:
   1. Sanitize the input
   2. Encrypt the incoming password using md5() method.
   3. In case any of the fields are empty print an error.
   4. If there is no error, we need to make a query into the users table, by the email and the encrypted password. Check eloquent to be able to have inside of $user the corresponding User object.
   5. After, in case we have a user, let´s save some information in the $_SESSION:

```
if (($user)) {
    $_SESSION['is_logged_in'] = true;
    $_SESSION['user_data'] = array(
        "id" => $user->id,
        "name" => $user->name,
        "email" => $user->email,
    );
    header('Location: ' . ROOT_URL);
} else {
    Messages::setMsg('Incorrect Login', 'error');
}
```

1. Managing Authentication

8. What we are missing is to be able to see if the user is logged in or not:
   1. In order to do so, go to your layout.html.file and create a condition that checks the is_logged_in variable from the $_SESSION. In case it is set, show “Welcome, Ines”, where Ines would be the name of the user. And a ‘Logout’ link.
      - That link should go to …/users/logout
   2. In case it is not logged in, show the previous “Login” and “Register” labels.

Test if the flow works when the user is correctly logged in, and when not. Check also the corresponding labels in the layout.

1. Managing Authentication

9. Finally, we are missing the *logout* functionality:
    1. Complete the logout method from the controller, this way:
        - Unsetting the $_SESSION variables.
        - Destroying the session
        - Redirecting the user to the home page

Follow this test:

1. Log in.
2. Check the cookie left in your explorer.
3. Check where is the session information saved in your server.
4. Log out and check again the cookie and the session information in your server.
5. Where is the *session_start* method called? Whenever you want to wark with sessions it must be called!!

2. Session messages

- These are messages showed to the user only once. They are also called *Flash messages*.
- For instance:
  - If the user is not correctly logged in→ "Incorrect Login".
  - In case a registration could not be succeded, a message can be shown like "Please Fill In All Fields".

2. Session messages

- Follow these steps

1. The first thing we need is the Messages class, which will be part of our core (remember the autoload):

```php
<?php
class Messages{
    public static function setMsg($text, $type){
        if($type == 'error'){
            $_SESSION['errorMsg'] = $text;
        } else {
            $_SESSION['successMsg'] = $text;
        }
    }

    public static function display(){
        if(isset($_SESSION['errorMsg'])){
            echo '<div class="alert alert-danger">'.$_SESSION['errorMsg'].'</div>';
            unset($_SESSION['errorMsg']);
        }

        if(isset($_SESSION['successMsg'])){
            echo '<div class="alert alert-success">'.$_SESSION['successMsg'].'</div>';
            unset($_SESSION['successMsg']);
        }
    }
}
```

2. Session messages

2. Setting Flah messages in our Users model:
   - *register* method:

```
19          if ($post['name'] == '' || $post['email'] == '' || $post['password'] == '') {
20              Messages::setMsg('Please Fill In All Fields', 'error');
21              return;
22          }
```

   - *login* method: on your own… in case the fields are not filled.
   - *login* method: in case the $user is empty:

```
62          if (($user)) {
63              $_SESSION['is_logged_in'] = true;
64              $_SESSION['user_data'] = array(
65                  "id" => $user->id,
66                  "name" => $user->name,
67                  "email" => $user->email,
68              );
69              header('Location: ' . ROOT_URL);
70          } else {
71              Messages::setMsg('Incorrect Login', 'error');
72          }
```

2. Session messages

2. Setting more Flah messages in our Books model:
   - *add* method: on your own… in case the fields are not filled →'Please Fill In All Fields'
3. Now, we need to print out those flash messages:
   - *Inside of our layout.html.php, add an static call to the display method from Messages and echo its returning value. Write the call exactly abouve the <?=$content?>*

Complete this test:

1. Incorrect log in.
2. Incorrect registration.
3. Incorrect POST add.

2. Session messages

Complete this functionality on your own:

1. Show a message to the user in case the user correctly logs in→'You are now logged in'
2. Inside of the index.html.php if the user is not logged_in the "New book" button won´t appear.
3. The same for the delete actions.
4. Morever, if the user tries adding a book (manually in the URL) and the user is not logged_in, he will not be able to add any books:
    1. Show him a message "You are not allowed to do this action"
    2. Redirect the user to the home page.
5. If the user tries deleting a book and is not logged in:
    1. Show him a message "You are not allowed to do this action"
    2. Redirect the user to the home page.

3. More about Sessions and Cookies

Save this session information:

1. You have to keep and show the last time the user was connected.
2. Add a “Remember me” checkbox to the login form. If the box is checked and the login is successful, save a cookie that identifies the user to the server. You may choose a reasonable expiration time for the cookie (one month). On further visits to the page, the user should appear logged in, even if the browser has been closed. You may choose a reasonable expiration time for the cookie. Remember also that if the user manually logs out by clicking the “Log Out” button that the cookie should be deleted.