DB Practical Work 1:
The User model

December 12, 2016

Abstract

The following subject aims at implementing the data handling for users in a twitter-like web-application. Implementations are to be done in the file `model/user.php`

Contents

1 The User entity

1.1 Presentation

The User entity represents a user and its properties:

- its login username (used in URLs and during identification)
- its displayed name (which is a name to be displayed in the application
- its hashed password (for identification)
- its email
- its avatar (or profile picture)

1.2 `create($username, $name, $password, $email, $avatar_path)`

This function inserts a user in database. *It is to be noted that the password must be hashed (see `hash_password($password)`)*

The function returns either the id of the newly inserted user. If there was a problem during the insertion, The `null` value is returned.

It doesn't check whether the username is already taken or not.

1.3 `get($id)`

This function gets a post with a given id (the one given in parameter).

The application asks for a particular output: `get($id)` must return a stdClass PHP object. Such an object can be declared as follows:

```
$o = (object) array(
"attribute" => "value"
);
```

In the case of our User entity, an object will be owning the following attributes:

```
$o = (object) array(
    "id" => 1337,
    "username" => "yrlgtm",
    "name" => "User 1",
    "password" => "hashed",
    "email" => "yrlgtm@gmail.com",
    "avatar" => "images/sddfvjdfvj.png"
);
```

1.4 `modify($uid, $username, $name, $email)`

This function updates a user whose id is `$uid`. It doesn't check whether the new username is already taken or not. It returns a boolean which gives the state of the query.

1.5 `change_password($uid, $new_password)`

This function updates only a user's password. This function hashes the new password. It returns a boolean.

1.6 `change_avatar($uid, $avatar_path)`

This function changes the avatar of the user. The provided path is a temporary path and therefore the file has to be saved somewhere else. Returns a boolean.

1.7 `destroy($id)`

This function deletes a user entry. It returns a boolean.

1.8 `search($string)`

This function searches for users by query on both username and displayed name.

1.9 `list_all()`

This function returns an array of every users objects (same return format as in `get($id)`).

1.10 `get_by_username($username)`

This function returns a user matching the given username (same return format as in `get($id)`). Returns null if no user were found.

1.11 `check_auth($username, $password)`

Tries to authenticate a username with a given password. Returns the user object (same return format as in `get($id)`) if everything went fine. Returns `null` else. *This function **does** need to hash the password*

1.12 `check_auth_id($id, $password)`

Tries to authenticate a user id with a given password. Returns the user object (same return format as in `get($id)`) if everything went fine. Returns `null` else. *This function **doesn't** need to hash the password*

2 Following users

2.1 Presentation

Users can follow each others. Following someone enables a user to receive in their timeline the other user's posts.

- A user's followers are the users following him/her
- A user's following are the users he/she follows

2.2 `follow($id, $id_to_follow)`

This function creates a "follow" association between two users. Returns a boolean.

2.3 `unfollow($id, $id_to_unfollow)`

This function deletes a "follow" association between two users. Returns a boolean.

2.4 `get_followers($uid)`

This function returns an array of objects for every users that follow a given user.

2.5 `get_followings($uid)`

This function returns an array of objects for every users that a given user follows.