

PHP Users Class

I got frustrated trying to find a class or library to authenticate user logins in PHP. The ones I found were either too simplistic, or required me to “join” some project just to have access to purportedly open source software.

So, I decided to dust off the neurons and see if I could build one. I decided to make it as flexible as possible, with only the very basics, but able to be enhanced via data calls. I also decided to make the data access independent of the class itself so data access classes could be (re)written for tasks other than MySQL using the mysqli library.

By itself, the users class (with a data access class `usersDataSource` like the included `UsersDataSourceMySQLi` class) handles basic login/logout/editing functions.

The class(es) were, however, designed for extensibility and customization in mind. Some design considerations were made with this in mind.

NOTE: This code uses the ternary and null coalescing shortcuts `?:` and `??`. I THINK these were introduced in PHP 5.3, but not sure. This code will not work on versions which do not have these shortcuts. See <https://www.php.net/manual/en/language.operators.comparison.php>

You can get a copy of this from our subversion repository

```
svn co http://svn.dailydata.net/svn/php_users/stable php_users
```

My working copy is at http://svn.dailydata.net/svn/php_users/trunk but I recommend NOT using that as I use trunk as my personal playground and will commit broken code to it regularly

Basic System

With no modification, the system will store username, password and two booleans, `isAdmin` and `enabled`. The default table is created as

```
CREATE OR REPLACE TABLE _users (  
  _user_id    INT UNSIGNED NOT NULL AUTO_INCREMENT,  
  login       VARCHAR(64),  
  password    VARCHAR(128),  
  isAdmin     BOOLEAN,  
  enabled     BOOLEAN,  
  PRIMARY KEY (_user_id)  
);
```

- login is filtered to alpha-numerics and the underscore character
- password is stored as a hash using PHP's `password_hash`
- users with `isAdmin` set will be able to add/edit other users
- users with `enabled` set to false (0) will not be able to log in

NOTE: the `usersDataSource` class has a public function, `buildTable`, which will build the table, so

installation involves simply calling that function.

IMPORTANT: to allow the Users class to work with a wide variety of data types, it does no data access itself. It requires a data access class.

Basic use in a script involves instantiating a data access class object, then instantiating a Users class object.

Example:

```
<?php
include_once( 'UsersDataSourceMySQLi.class.php' );
include_once( 'Users.class.php' );
session_start();
$connection = new usersDataSource(
    null,
    null,
    array( 'username' => 'test', 'password' => 'test', 'database' =>
'test' )
);
// $connection->buildTable( 'admin', 'admin' ); die;
// ensure we always have a (possibly invalid) instance of user
if ( ! isset( $_SESSION['user'] ) ) {
    $_SESSION['user'] = new Users( );
}
?>
<html>
  <body>
    <div class="login">
      <?php
        if ( isset( $_SESSION['user'] ) )
          print $_SESSION['user']->HTML($connection);
      ?>
    </div>
  </body>
</html>
```

This example is using the usersDataSource definition of data access (included)

If you run it the first time with

```
$connection->buildTable( 'admin', 'admin' ); die;
```

uncommented, it will build the table. Comment that line out on the next run and you will be presented with a login screen.

Class function HTML() displays various things to allow login, then quits displaying anything. Setting `$_REQUEST['logout'] = 1` before calling HTML() will initiate a log out which will destroy the session variable

Full Functionality

Calling the class method `admin` and displaying the repeatedly will generate output allowing the user to change their username, password and any other fields which do not have the 'restrict' attribute set to true

If the user has admin rights set, it will also display a list of logins and allow you to select one to edit or, add a new user. Editing someone else shows all fields, whether or not the 'restrict' attribute is set to true.

CSS

I tried to not put any HTML layout into the code, relying instead on CSS. Everything is supposed to have a class and be wrapped in a `<div>` with a class. Following are the classes I have in the code (I THINK). * `login_field` = This is the class of all INPUT fields, and div's surrounding them * `login_form` = the class for all `<form>` definitions * `login_list` = the class of the unsigned list (ul) which displays the links to edit other users for admin's * `login` = NOT in code, but used in example as a div around the div around the login area

Extended Functionality

First, everything is pretty basic. I tried to limit the number of fields to the absolute minimum, but also set it up to allow additional fields to be added programmatically. The example does this.

If you open the class source, you'll find the private member `$dbDefinition`, which defines everything in the code (I hope). When you create a new instance, you can pass the constructor an array which will be merged with this member, optionally increasing the number and definition of the fields.

WARNING: if you add new fields to the Users class, you must also add them to class `usersDataSource`. See below

Let's add a new field, say we want to store the users e-mail address. In our PHP, create an array as follows:

```
$customFields = array(
    'tables' => array(
        'users' => array(
            'fields' => array(
                'email' => array(
                    // == For Users class ==
                    // this will be the display label on the form
                    'label' => 'E-Mail',
                    // the input type to use for data entry
                    'html type' => 'text',
                    // you can only edit this if an admin and changing
someone
                    // else' record
                    'restrict' => false,
```

```

// will be displayed on a hover in HTML5 (ie, title=)
'instructions' => 'Enter your e-mail address (WARNING,
not verified)',

// this is entered in an empty box, ie placeholder=
'hint'      => 'E-Mail Address',
// a regex to run it against to verify it is ok
'filter'    => '/[-_a-z0-9.]+@[_a-z0-9]\.[a-z0-9]/i',

// == for Data Source ==
'dbColumn'  => 'email',
// actual mySQL column type
'type'      => 'varchar(64)',
// set it to not null if we build the table ourselves
'required'  => false
)
)
)
);
?>

```

Now, when we instantiate a new object of class Users AND class usersDataSource, we simply pass this array in.

<code php>

```

$connection = new usersDataSource(
    null,
    $customFields,
    array( 'username' => 'test', 'password' => 'test', 'database' =>
'test' )
);
if ( ! isset( $_SESSION['user'] ) ) {
    $_SESSION['user'] = new Users( $customFields );
}

```

</php>

Note that since we replicated the basic structure of \$dbDefinition in Users and usersDataSource, we can use the same hash to pass into both; they will store, but ignore values not relevant to them.

When the usersDataSource and Users objects are created, \$customFields will be merged, with duplicates overwritten by the \$customFields value.

This is not limited to adding new columns; you can modify the display definitions also, ie how the information is stored on the screen, to some extent.

usersDataSource

This is our data access class. It really doesn't matter what it is called, though I plan to call it the same when I add more data access objects.

This code accesses the data (duh), and is consistently called `$connection` in the Users class. The only requirement is that it must be able to implement the following functions

`getPassword( $username )` returns encrypted password
`getRecord( $username )` returns array containing the values for a user
`getAllUsers()` returns an array of all user id's and names
`getARecord( array of key value pairs to limit what is retrieved )` returns all values
`update( array of key value pairs )` returns true/false on success/failure

It is also nice is it can A) handle new columns and B) create and initialize the necessary storage

I separated this out from the Users class because not all programs need database access. For instance, the favorites_urls app uses file based storage, so by writing a new access class for it, we will hopefully be able to get the same functionality, but with a different storage back end.

Future

This is only the initial part of this particular project. I now intend to extend both classes to allow boolean permissions which will be integrated into our new version of CAMP, giving very granular rights to users. It will be available as a second set of files in this repository and is planned for release by October 2021.

From:

<https://kb.unixservertech.com/> - **Unix Server Tech Knowledge Base**

Permanent link:

https://kb.unixservertech.com/software/dailydata/libraries/php_user?rev=1629928464

Last update: **2021/08/25 21:54**

