
Bs_Cug:

closed user groups for everyone :-)

Andrej Arn
Sam Blume



BlueShoes

THIS IS A STEP-BY-STEP GUIDE TO CREATE A CLOSED USER GROUP (MEMBER SECTION, PASSWORD-SECURED AREA) USING A MYSQL DATABASE. WE USE BS_CUGDB WHICH EXTENDS BS_CUG.

Featur:s:

- hack detection
- anti hack mechanism (Unix-like doubling of the timeout)
- built-in account lifecycle
- logging (with automatic db-table generation)
- checking case sensitive or not for usernames and passwords
- registered [additional] custom validators

Source Locations:

core/auth/cug/Bs_Cug.class.php
core/auth/cug/Bs_CugDb.class.php
core/auth/cug/examples/cugdb/index.php and cug.inc.php [and create.sql]

What you need:

- An existing installation of PHP/MySQL/Apache (or IIS) and *BlueShoes*.
- A website running *BlueShoes*.
- Some knowledge of PHP, understanding of how to use existing classes. Some knowledge of how to use SQL/MySQL.

The example source files may be helpful if you get stuck or to copy/paste.

Create a new directory in your webroot, in my case that's: /examples/Cug/

In that directory create an empty file with the name cug.inc.php.

Part 1:

```
<?php
require_once($_SERVER["DOCUMENT_ROOT"] . "../global.conf.php");
require_once($APP['path']['core'] . 'auth/cug/Bs_CugDb.class.php');
$cug =& new Bs_CugDb('myExampleCug');
$cug->language = 'en';
$cug->userDbName = 'test';
$cug->userTableName = 'ExampleCug';
$cug->logDbName = 'test';
$cug->logTableName = 'ExampleCugLoginLog';
```

Include the needed files. Then create a new instance. The CUG's name is 'myExampleCug'. We use two db tables. One where we manage and look up the users, and one to log all login attempts. Change these settings for your needs.

Part 2:

```
$dsn = array(
    'name'=>'test', 'host'=>'localhost', 'port'=>'3306', 'socket'=>'',
    'user'=>'root', 'pass'=>'', 'syntax'=>'mysql', 'type'=>'mysql'
);
if (isEx($bsDb =& getDBObject($dsn)) $dbAgent->stackDump('die');
$cug->setDBObject($bsDb);
```

We need a connection to the db. Maybe you create one in your global.conf.php already, if so, you can use that one. Then attach it to the cug using `setDBObject()`. Customize the dsn for your needs.

Part 3:

```
if (isset($_GET['logout'])) {
    $cug->logout();
    $outMsg = "<font color='green'><b>You have logged out successfully.</b></font>";
}
```

It is wise to allow the user to log out. This way one can log in with a different name, or leave the computer. Just add a link to `"/?logout=1"` somewhere.

Part 4:

```
if (isset($_GET['logout']) || !$cug->letMeIn()) {
    $loginForm = $cug->treatLoginForm();
    if ($loginForm === TRUE) {
        //oki doki, logged in successfully, register some vars.
        //getFieldValue returns a vector with exactly one element.
        $t = $cug->_form->getFieldValue('username');
        $username = $t[0];
        $cug->bsSession->register('user', $username);
        redirect('http://developer.blueshoes.org/index.php'); //script ends here.
    }
?>
```

If the user just logged out, or is not logged in already (`letMeIn()`) then we have to deal with the login form. There are two options. We have to display the form, or we have to read the submitted form. Whatever it is, `treatLoginForm()` handles this for us. All we have to do is check the return value. If it was a successful login, then we register the username into the session. And then we redirect to the frontpage. Change the URL for your needs.

Part 5:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>
<head>
  <title>BlueShoes - Closed User Group</title>
  <link href="http://www.blueshoes.org/default.css" rel="stylesheet">
</head>

<body leftmargin="0" topmargin="0">

<table border='0' cellpadding='0' cellspacing='0' width='100%' height='100%'>
  <tr><td bgcolor='#212B43' height='20'>&nbsp;</td></tr>
  <tr><td bgcolor='white' height='1'></td></tr>
  <tr><td bgcolor='#2E3C5E' height='50'>&nbsp;</td></tr>
  <tr><td bgcolor='white' height='1'></td></tr>
  <tr>
    <td align="center" valign="middle">
      <table width='350'><tr><td>
        <h2>closed user group</h2>
        <?php
          if (isset($outMsg)) {
            echo $outMsg . "<br><br>";
          }
          echo $loginForm;
        ?>
      </td></tr></table>
    </td>
  </tr>
  <tr><td bgcolor='white' height='1'></td></tr>
  <tr><td bgcolor='#2E3C5E' height='50'>&nbsp;</td></tr>
  <tr><td bgcolor='white' height='1'></td></tr>
  <tr><td bgcolor='#212B43' height='50'>&nbsp;</td></tr>
</table>

</body>
</html>
<?php
  exit;
}
?>
```

This is pretty much only the html page. It has some PHP in the body and at the end. It's the login screen, and the login form is spitted out in the middle. The Bs_Form package is used for that.¹

Now create a new empty file in the same directory with the name index.php.

```
<?php
include './cug.inc.php';
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>
<head>
  <title>Untitled</title>
</head>

<body>

hello <?php echo $cug->bsSession->getVar('user');?>
<br><br><br><br>

<a href="<?php echo $_SERVER['PHP_SELF'];?>?logout=1">logout</a><br>

</body>
</html>
```

This is a very simple example page of the cug. Like all scripts that have to be secured by the cug have to include the cug.inc.php on top. Later down this script welcomes the logged in user. And then there's a link to log out.

That's it from the code side. What's missing? Right... the database.

The logins are managed in a table. So we need a table first. In the code we specified the name "ExampleCug".

```
CREATE TABLE ExampleCug (
  ID          INT UNSIGNED NOT NULL AUTO_INCREMENT DEFAULT 0,
  user        VARCHAR(20)  NOT NULL DEFAULT '',
  pass        VARCHAR(20)  NOT NULL DEFAULT '',
  isActive    TINYINT      NOT NULL DEFAULT 0,
  startDatetime DATETIME   NOT NULL DEFAULT '0000-00-00 00:00:00',
  endDatetime  DATETIME   NOT NULL DEFAULT '0000-00-00 00:00:00',
  PRIMARY KEY ID (ID),
  KEY user (user)
);
```

ID	is the unique identifier of the table.
user	is the username.
pass	is the password.
isActive	is a flag that tells if the account is active or not. 0=inactive, 1=active.
startDatetime	if used, the account will become active on that datetime. But only if isActive is 1, not 0.
endDatetime	if used, the account will expire on that datetime.

And then we want 3 different logins.

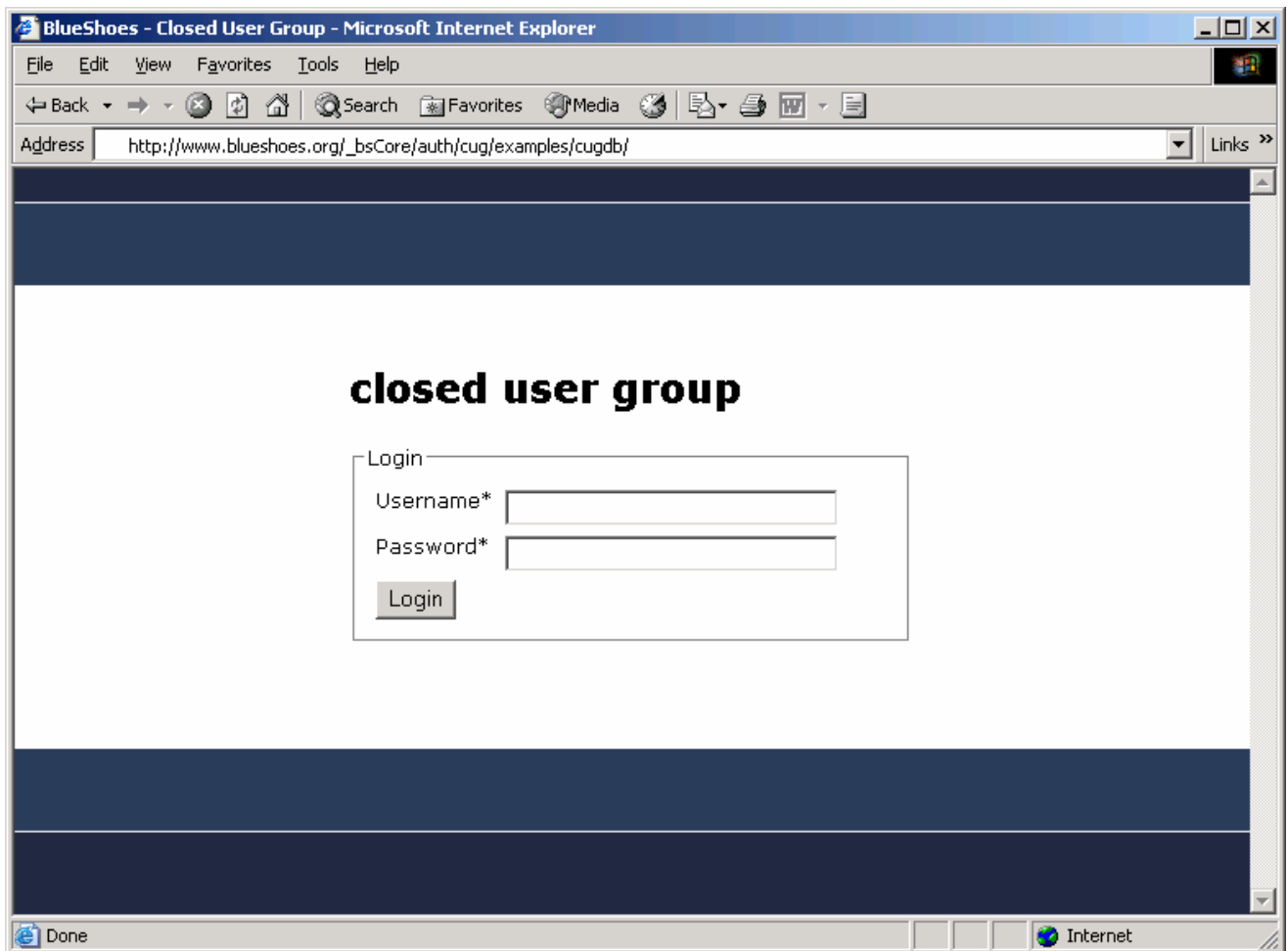
```
Insert Into ExampleCug (user, pass, isActive, startDatetime, endDatetime) VALUES
('alanis', 'morisette', 1, '', '');

Insert Into ExampleCug (user, pass, isActive, startDatetime, endDatetime) VALUES
('bill', 'gates', 0, '2090-01-01 12:00:00', '');
```

```
Insert Into ExampleCug (user, pass, isActive, startDatetime, endDatetime) VALUES  
( 'student', 'holiday', 1, '2003-10-25 00:00:00', '2004-11-30 23:59:59' );
```

The account for alan is active right away. Bill's account would become active on January 1 2090, but it's set to inactive. Student's account will become active on 2003/10/25 and will automatically expire on 2004-11-30.

That was it. Point your browser at the new url, in my case that is http://www.blueshoes.org/_bsCore/auth/cug/examples/cugdb/.



Additional notes:

- If the db table does not have the fields `isActive`, `startDatetime` and `endDatetime` then the lifecycle feature will be ignored.
- To prevent simultaneous logins with the same account, check `Bs_CugDb->killOldSessions()`.
- To use additional login validation code, check `Bs_Cug->registerValidator()`.
- `Bs_Cug` uses an instance of `Bs_SimpleSession`.²
- To manipulate the look/behavior of the login form, use `Bs_Cug->loadLoginForm()`, and then access `$cug->form` (instance of `Bs_Form`¹) directly.
- To use hack detection (`Bs_Cug->looksLikeHack()`) you need the knowledge base with the bad usernames/passwords.

References

- 1 `Bs_Form`, see <http://www.blueshoes.org/en/framework/html/form/>
- 2 See the class `core/net/http/session/Bs_SimpleSession.class.php` and the HOWTO http://www.blueshoes.org/howtos/Bs_SimpleSession.howTo.pdf