

```

<?php
/**
 * The user class is intended to handle information about authenticated users. Information
 * contained in this class is to be serialized and stored in $_SESSION['user_data']
 *
 * Author: WigeDev
 * Version: 1.0
 * Created: June 29, 2010
 * Modified: August 25, 2010
 */
class User {

    private $_username = '';
    private $_userid = '';
    private $_userlevel = '';
    private $_name = '';
    private $_email = '';
    private $_authenticated = '';
    private $_useragent = '';
    private $_ip = '';

    /**
     * The constructor sets the variables to default values by calling the reset function
     */
    function __construct() {

        $this->reset();

    }

    /**
     * When the user is unserialized, check the IP address and user agent. If both have changed
     * since the object was initialized, reset the object.
     */
    function __wakeup() {

        if ($this->_useragent != $_SERVER['HTTP_USER_AGENT'] && $this->_ip != $_SERVER['REMOTE_ADDR']) {
            $this->reset();
        }

    }

    /**
     * The reset function sets the member values back to default. This is to be used when
     * the class is initialized, or if the user fails a reauthentication process.
     */
    private function reset() {

        $this->_username = NULL;
        $this->_userID = NULL;
        $this->_userlevel = 'guest';
        $this->_name = NULL;
        $this->_email = NULL;
        $this->_authenticated = FALSE;
        $this->_useragent = $_SERVER['HTTP_USER_AGENT'];
        $this->_ip = $_SERVER['REMOTE_ADDR'];

    }

    /**
     * The authentication function gets the user information from the database according to the
     * username and password provided.
     *
     * @param inUsername The provided username
     * @param inPassword The provided password
     * @returns True if the user authenticated successfully, false otherwise
     */

```

```

function authenticate($inUsername, $inPassword) {

    if (validateUsername($inUsername) === FALSE) throw new BadAuthentication('username invalid');
    if (validatePassword($inPassword) === FALSE) throw new BadAuthentication('password invalid');
    $username = $inUsername;
    $password = $inPassword;

    // Get the user information from the database
    $query = "SELECT id, username, name, email FROM users WHERE username = '$username' AND password
= PASSWORD('$password') LIMIT 1";
    $result = mysql_query($query);
    $data = mysql_fetch_assoc($result);
    echo mysql_error();

    // Check that the data returned is valid
    if (isset($data['id'])) {
        // The user authenticated, check that the account is active
        $this->_username = $data['username'];
        $this->_userID = $data['id'];
        $this->_email = $data['email'];
        $this->_name = $data['name'];
        $this->_userlevel = 'user';
        $this->_authenticated = TRUE;
    } else {
        // The user could not be authenticated
        throw new BadAuthentication('password');
    }

}

function getUsername() {
    return $this->_username;
}

function getID() {
    return $this->_userID;
}

function getName() {
    return $this->_name;
}

function getLevel() {
    return $this->_userlevel;
}

function isAuthenticated() {
    return $this->_authenticated;
}

/**
 * Function tests the passed username to verify that the username only contains letters and
 * numbers.
 *
 * @param data The submitted username.
 * @return True if the username passes validation, False otherwise
 */
private function validateUsername($data) {
    if (preg_match("/^[A-Za-z0-9]+$/", $data) != 1) {
        return FALSE;
    } else {
        return TRUE;
    }
}

/**
 * Function checks the submitted password to verify that it only contains allowed characters,

```

```
* specifically letters, numbers, and certain punctuation characters.
*
* @param data The submitted password
* @return True if the password contains only valid characters, False otherwise
*/
private function validatePassword($data) {
    if (preg_match("/^[A-Za-z0-9\!\@\#\$\&\-]+$/", $data) != 1) {
        return FALSE;
    } else {
        return TRUE;
    }
}

}

/**
 * The BadAuthentication exceptions are thrown when authentication fails. Generally this will be
 * caused by problems with the passed username/password combination.
 */
class BadAuthentication extends Exception {

}
?>
```