

In my previous article [Write an Advanced Joomla 1.5 Authentication Plugin](#) I went over how to integrate Joomla with an alternate login system (i.e. legacy code from a previous project)

This is good, but we don't need to stop here. We can improve the authentication plugin by attaching session data from our legacy system and essentially merge our legacy session with our Joomla session.

For example if we have a database table for session data with the following fields:

```
session_id : int PRIMARY KEY
username   : varchar(20)
address    : varchar(255)
city       : varchar(255)
state      : varchar(2)
...
```

To append this data into our Joomla session we just add a function to listen to the onLoginUser event in our authentication plugin.

```
/*
 * onLoginUser() occurs after the user has authenticated and is the last
 * event that gets called in the login process
 *
 * @param $credentials username and password sent from login form
 * @param $remember     whether or not the checkbox "remember me" was
checked
 */
function onLoginUser( $credentials, $remember) {
    $username = mysql_real_escape_string($credentials['username']);
    $sql = "SELECT * FROM session_data WHERE `username`='$username' LIMIT 1";
    $result = mysql_query( $sql );
    if (!$result) {
        die( "No result for user." . $sql );
    }
    $user = mysql_fetch_assoc($result);
```

```
$session = JFactory::getSession();
$session->set('legacy_user', $user);

return;
}
```

What this code does is run an SQL query and assign the result to a key in our session.

To access our data later on (from anywhere inside joomla) we can do the following:

```
$session =& JFactory::getSession();
$legacy_session = $session->get('legacy_user');
echo $legacy_session['address'];
```

And here is our updated plgMyAuth.php file from our last article:

```
/**
 * @package Lab11.Tutorials
 * @subpackage Plugins
 * @license GNU/GPL
 */

// Check to ensure this file is included in Joomla!
defined('_JEXEC') or die();

// Make sure we have a JPlugin class to extend from
jimport('joomla.event.plugin');

/**
 * Example Authentication Plugin. Based on the example.php plugin in the
 * Joomla! Core installation
 *
 * @package Lab11.Tutorials
 * @subpackage Plugins
 * @license GNU/GPL
 */
class plgAuthenticationMyAuth extends JPlugin
{
    /**
     * Constructor
     *
     * For php4 compatability we must not use the __constructor as a constructor
```

```

* for plugins because func_get_args ( void ) returns a copy of all passed
* arguments NOT references. This causes problems with cross-referencing
* necessary for the observer design pattern.
*
* @param object $subject The object to observe
* @since 1.5
*/
function plgAuthenticationMyAuth(& $subject) {
    parent::__construct($subject);
}

/**
* This method should handle any authentication and report back to
* the subject
*
* @access public
* @param array $credentials Array holding the user credentials
* @param array $options Array of extra options
* @param object $response Authentication response object
* @return boolean
* @since 1.5
*/
function onAuthenticate( $credentials, $options, &$response )
{
    /*
    * Here you would do whatever you need for an authentication routine
    * with the credentials
    *
    * In this example the mixed variable $return would be set to false
    * if the authentication routine fails or an integer userid of the
    * authenticated user if the routine passes
    */

    // grab user/pass from $credentials passed in from Joomla's login form.
    $username = mysql_real_escape_string($credentials['username']);
    $password = mysql_real_escape_string( md5($credentials['password']) );

    // build our query to retrieve a user account
    $sql = "SELECT * FROM `alternate_users` WHERE `username`='<
    &quot;and `password`='<
    &quot; LIMIT 1";
    // run the query
    $result = mysql_query( $sql );
    if (!$result) {
        // ... for extended debugging info, uncomment the following line.
        //die(&quot;No result for user.&quot;;. $sql . &quot;; Errors: &quot;; . mysql_error()) ;
    }
}

```

```

        $response->status = JAUTHENTICATE_STATUS_FAILURE;
        $response->error_message = &quot;Unable to query database.&quot;;
        return false;
    }

    // get our user data from
    $user = mysql_fetch_assoc($result);

    if (empty($user)) {
        $response->status = JAUTHENTICATE_STATUS_FAILURE;
        $response->error_message = &quot;Unable to find user&quot;;
        return false;
    }

    // once we have our user, pass some data to joomla
    $response->email = $user['email'];
    $response->fullname = $user['contact'];
    $response->status = JAUTHENTICATE_STATUS_SUCCESS;

    // return successfully
    return true;
}

/**
 * This method alters our alternate table to reference our
 *
 * @access public
 * @param array $columns An array of the columns in the user table.
 * @param object $id A unique identifier for this user.
 * @return none
 * @since 1.5
 */
function onAfterStoreUser( $columns, $id )
{
    $user = mysql_real_escape_string($columns['username']);
    $user_id = mysql_real_escape_string($columns['id']);

    $sql = &quot;UPDATE `l11m_cgl_users` SET joomla_uid = '&quot; . $user_id .
&quot;'&quot;
        . &quot; WHERE username='&quot; . $user . &quot;'&quot;;
    mysql_query($sql) or die(&quot;failed to update user. &quot; . mysql_error());
}

/**
 * onLoginUser() occurs after the user has authenticated and is the last

```

```
* event that gets called in the login process
*
* @param $credentials  username and password sent from login form
* @param $remember    whether the checkbox 'remember me' was checked
*/
function onLoginUser( $credentials, $remember) {
    $username = mysql_real_escape_string($credentials['username']);
    $sql = "SELECT * FROM session_data WHERE `username`='$username' LIMIT
1";
    $result = mysql_query( $sql );
    if (!$result) {
        die( "No result for user.". $sql );
    }
    $user = mysql_fetch_assoc($result);

    $session = JFactory::getSession();
    $session->set('legacy_user', $user);

    return;
}
}
```