Subject: BUG+FIX: Problems with backslashes before qoutes in getdata($where)
Posted by janalwin on Mon, 18 Dec 2006 14:13:17 GMT
View Forum Message <> Reply to Message

I get an sql error when I do something like this: (it's testcode, so it doesn't make sense in a real situation)

$where="product_code='pc' AND user='testuser' AND test='test\\\\' AND active=1";
$x=$dbobject->getData($where);

There's an escaped bakslash directly before the quote (test='test\\').

I traced the problem back to the extractValue function in include.general.inc.

The extractValue function thinks that \\' means that the ' is escaped. The is not the case of course, the \ is escaped!.

I modified the extractValue function a bit:

- The function now searches for the delimiter or the delimiter with any number of backslashes before it.
- If the length of the found string is even, it has to be an escaped delimiter. So the function will search again to find the real delimiter.

ODD gives a not escaped qoute (delimiter), examples: ', \\', \\\\', etc...
EVEN gives a escpaped qoute, examples, \', \\\', \\\\\' etc..

I think this solves the problem. I hope someone can check my code for bugs and unexpected behaviour.

Here is the new code. Changes are marked with CHANGE JADJ

```
function extractValue (&$where)
// extract value from a WHERE string where the value is delimited by either
// single or double quotes.
// WARNING: any ending delimiter which is escaped must be ignored.
// NOTE: $where is passed by reference so that it can be modified
{

   // the first character is the delimiter (single or double quote)
   $delimiter = substr($where, 0, 1);
   if ($delimiter == "" or $delimiter == """ or $delimiter == ' ' or $delimiter == '(') {
      // delimiter is valid
   } else {
      // no valid delimiter found, so use a space instead
      $delimiter = ' ';
   } // if

   // look for ending delimiter
```

```
    if ($delimiter == """ or $delimiter == "") {
        // delimiter may be escaped with preceeding '\'

        // CHANGE JADJ
        //$pattern = '/(\\\\' .$delimiter .'|' .$delimiter .')/';
        $pattern = '/(\\\\+' .$delimiter .'|' .$delimiter .')/';

         } elseif ($delimiter == ' ') {
        $pattern = "/ /";
    } else {
        // look for closing parenthesis
        $pattern = "/\)/";
    } // if

    if (preg_match($pattern, $where, $regs, PREG_OFFSET_CAPTURE, 1)) {
        $found  = $regs[0][0];
        $endpos = $regs[0][1];


        // CHANGE JADJ
        // check if length of found is odd or even..
        $odd_found_len=strlen($found) & 1;

        // CHANGE JADJ
        // length of found has to be odd if it's the end of the fieldvalue
        // So continue searching...
        while (!$odd_found_len) {
            // found an escaped delimiter, so ignore it
            // CHANGE JADJ
            preg_match($pattern, $where, $regs, PREG_OFFSET_CAPTURE, $endpos+
strlen($found));
            $found = $regs[0][0];
            $endpos = $regs[0][1];

            // CHANGE JADJ
            $odd_found_len=strlen($found) & 1;

        } // while


        // CHANGE JADJ
        // if we had a match we need another way to get the fieldvalue than in the old version
        // extract the string portion which exists between the two delimiters
        $fieldvalue = rtrim(substr($where, 0, $endpos + strlen($found)));
    } else {
        // not found so....
        if ($delimiter == '(') {
            // add missing delimiter
```

```
        $where .= ')';
       } // if
       // extract remainder of string
       $endpos = strlen($where);

       // CHANGE JADJ
       // Not found. We have to use the old way to get the fieldvalue
       // extract the string portion which exists between the two delimiters
       $fieldvalue = rtrim(substr($where, 0, $endpos + 1));
    } // if



    if (substr($fieldvalue, -1) == ')') {
       // last character is ')' so first character must be '('
       if (substr($fieldvalue, 0, 1) != '(') {
        // no match found, so remove trailing ')'
          $fieldvalue = substr($fieldvalue, 0, -1);
       } // if
    } // if
    // remove $fieldvalue from the front of the string
    $where = substr($where, strlen($fieldvalue));

    return $fieldvalue;

} // extractValue
```

Subject: Re: BUG+FIX: Problems with backslashes before qoutes in getdata($where)
Posted by AJM on Mon, 18 Dec 2006 16:12:39 GMT
View Forum Message <> Reply to Message

This is a very obscure problem that will only affect those who have a string field which terminates with a backslash, as in 'test\', which is most unusual. I have tried it with 'test\test' and this does not cause a problem.

I will take a look at your code and include a fix in the next release.

Subject: Re: BUG+FIX: Problems with backslashes before qoutes in getdata($where)
Posted by janalwin on Mon, 18 Dec 2006 16:28:00 GMT
View Forum Message <> Reply to Message

I agree that it is an obscure problem. This is how i discovered it:

I use parts of the framework for the frontend of a website. I use the database with validation, not the template part. I use the complete framework for the backend of the site.

A user tried to log in to the frontend of de site and made a typo in his username. He accidentally typed a backslash at the end of his username. I got a mail with a strange error from the system and started to investigate the problem....

The system is running for a few months now and this error has never occured before.

--
Jan Alwin de Jong

Subject: Re: BUG+FIX: Problems with backslashes before qoutes in getdata($where)
Posted by AJM on Mon, 18 Dec 2006 16:42:48 GMT
View Forum Message <> Reply to Message

It's just like a ham-fisted user to case a problem.

I can test for many things, but the old chair-to-keyboard-interface gets me every time. Oh well, back to the drawing board.