mysql_real_escape_string

(PHP 4 >= 4.3.0, PHP 5, PECL mysql:1.0)

mysql_real_escape_string — Maskiert spezielle Zeichen innerhalb eines Strings für die Verwendung in einer SQL-Anweisung

Beschreibung

string **mysql_real_escape_string** (string `$unescaped string` [, resource `$link identifier`])

Maskiert spezielle Zeichen im *unescaped_string* unter Berücksichtigung des aktuellen Zeichensatzes der Verbindung, so dass das Ergebnis ohne Probleme in mysql_query() verwendet werden kann. Wenn Sie Binärdaten einfügen wollen, müssen Sie die Funktion auf jeden Fall verwenden.

mysql_real_escape_string() ruft die Funktion mysql_real_escape_string der MySQL-Bibliothek auf, die folgende Zeichen mit einem Backslash ('\') versieht: *\x00*, *\n*, *\r*, **, *'*, *"* und *\x1a*.

Die Funktion muss immer (mit wenigen Ausnahmen) verwendet werden, um Daten abzusichern, bevor sie per Query an MySQL übermittelt werden.

Parameter Liste

unescaped string

Der zu maskierende String.

Verbindungs-Kennung

Die MySQL-Verbindung. Wird die Verbindungskennung nicht angegeben, wird die letzte durch mysql_connect() geöffnete Verbindung angenommen. Falls keine solche Verbindung gefunden wird, wird versucht, eine Verbindung aufzubauen, wie es beim Aufruf von mysql_connect() ohne Angabe von Argumenten der Fall wäre. Falls zufällig keine Verbindung gefunden oder aufgebaut werden kann, wird eine Warnung der Stufe **E_WARNING** erzeugt.

Rückgabewerte

Gibt einen maskierten String oder im Fehlerfall **FALSE** zurück.

Beispiele

Example#1 Einfaches mysql_real_escape_string()-Beispiel

```
<?php
// Verbindung herstellen
$link = mysql_connect('mysql_host', 'mysql_user', 'mysql_password')
    OR die(mysql_error());

// Anfrage erstellen
$query = sprintf("SELECT * FROM users WHERE user='%s' AND password='%s'",
            mysql_real_escape_string($user),
            mysql_real_escape_string($password));
?>
```

Example#2 Ein beispielhafter SQL Injection Angriff

```
<?php
// Datenbankabfrage zur Ueberpruefung der Logindaten
$query = "SELECT * FROM users WHERE user='{$ POST['username']}' AND password='{$ POST['password']}'";
mysql_query($query);

// Wir haben $_POST['password'] nicht geprueft, es koennte also alles darin
// stehen, was der User will. Zum Beispiel:
$ POST['username'] = 'aidan';
```

```
$_POST['password'] = "' OR ''='";

// Das bedeutet, der an MySQL gesendete Query wuerde sein:
echo $query;
?>
```

Die Abfrage, die an MySQL übermittelt wird:

```
SELECT * FROM users WHERE user='aidan' AND password='' OR ''=''
```

Dies würde jedermann erlauben, sich ohne valides Passwort einzuloggen.

Example#3 Optimale Vorgehensweise zur Querybehandlung

Die Verwendung von **mysql_real_escape_string()** bei jeder Variablen beugt SQL Injection Angriffen vor. Das Beispiel demonstriert ein optimales Verfahren für Datenbankanfragen, das unabhängig vom für Magic Quotes gesetzten Wert funktioniert.

```
<?php

if (isset($_POST['product_name'])
    && isset($_POST['product_description'])
    && isset($_POST['user_id'])) {

 // Verbinden mit der Datenbank
    $link = mysql_connect('mysql_host', 'mysql_user', 'mysql_password')

    if(!is_resource($link)) {

        echo "Verbindung zum Server fehlgeschlagen\n";
     // ... den Fehler loggen

    } else {

     // Die Auswirkungen von magic_quotes_gpc/magic_quotes_sybase zurücksetzen,
     // sofern die Option auf ON gesetzt ist

        if(get_magic_quotes_gpc()) {
            $product_name        = stripslashes($_POST['product_name']);
            $product_description = stripslashes($_POST['product_description']);
        } else {
            $product_name        = $_POST['product_name'];
            $product_description = $_POST['product_description'];
        }

     // einen sicheren Query zusammenstellen
        $query = sprintf("INSERT INTO products (`name`, `description`, `user_id`) VALUES ('%s', '%s', %d)'
                    mysql_real_escape_string($product_name, $link),
                    mysql_real_escape_string($product_description, $link),
                    $_POST['user_id']);

        mysql_query($query, $link);

        if (mysql_affected_rows($link) > 0) {
            echo "Produkt eingefuegt\n";
        }
    }
} else {
    echo "Fuellen Sie das Formular korrekt aus.\n";
}
?>
```

Die Anfrage wird jetzt korrekt ausgeführt und SQL Injection Angriffe funktionieren nicht mehr.

Anmerkungen

Hinweis: Sie müssen eine Verbindung zu MySQL geöffnet haben, bevor Sie **mysql_real_escape_string()** verwenden, ansonsten erhalten Sie einen Fehler vom Typ *E_WARNING* und der Rückgabewert wird zu **FALSE**. Ist *link_identifier* nicht angegeben, wird die letzte MySQL-Verbindung verwendet.

Hinweis: Ist magic_quotes_gpc aktiviert, wenden Sie zuerst stripslashes() auf die Daten an. Das Bearbeiten bereits in irgend einer Form maskierter Daten durch mysql_real_escape_string führt ansonsten dazu, dass bereits Maskiertes doppelt maskiert wird.

Hinweis: Wenn die Funktion nicht verwendet wird, um die Daten zu maskieren, ist der Query anfällig für SQL Injection Angriffe.

Hinweis: **mysql_real_escape_string()** maskiert weder % noch _. Diese Zeichen werden in MySQL als Platzhalter interpretiert, wenn sie mit *LIKE*, *GRANT* oder *REVOKE* kombiniert werden.

Siehe auch

- mysql_client_encoding()
- addslashes()
- stripslashes()
- Die magic_quotes_gpc-Direktive
- Die magic_quotes_runtime-Direktive