Unicode und PHP <= 5

Dieser Beitrag muss noch weiter überarbeitet werden.

== Ursache des Problems ==
Zeichensätze sind ein Abenteuer. Verfolgt man mal die Abläufe vom Editor zur Bash, von der Bash zum Terminal, vom Terminal in den Webserver, vom Webserver zum PHP Prozess, der PHP Prozess liest die Datei die einen Zeichensatz aufweist, der Code verbindet sich dann über das Connection Encoding zum MySQL Prozess, davon zur MySQL Datenbank, von der MySQL Datenbank zur Tabelle und von der Tabelle zu den Spalten so möchte man kaum wahr haben, dass jeder dieser Schritte ein eigenes Encoding verwendet kann.

Aus Erfahrung kann man falsche Daten in der Datenbank haben, die aus irgendwelchen Gründen richtig beim Client landen.
Schalten man irgendwann neue Clients hinzu könnte der Fehler erst auffallen.

== Lösung des Problems ==
Daher ist es sehr wichtig auf einen einheitlichen Zeichensatz zu setzen.
Gegen des Quasi-Standard UTF-8 sprechen keine mir bekannten Argumente. Daher baut dieser Artikel darauf auf, sein System auf UTF-8 zu stellen.

== Datenbank umstellen ==
Quellcode

```
1. ALTER DATABASE db_name
2. CHARACTER SET utf8
3. DEFAULT CHARACTER SET utf8
4. COLLATE utf8_general_ci
5. DEFAULT COLLATE utf8_general_ci
6. ;
8. ALTER TABLE tbl_name
9. DEFAULT CHARACTER SET utf8
10. COLLATE utf8_general_ci
11. ;
```

Alles anzeigen

== Korrekte String Funktionen benutzen ==
In PHP 5 sollte man keine Standard-Stringfunktionen wie strlen, strpos, ... mehr nutzen.
Benutzt stattdessen die Multibyte String Funktionen wie sie hier zu finden sind: php.net/manual/de/ref.mbstring.php

Konkret könnt ihr euren gesamten Quelltext durchsuchen und folgende Funktionen ersetzen:
Quellcode

```
1. mail() -> mb_send_mail()
2. strlen() -> mb_strlen()
3. strpos() -> mb_strpos()
4. strrpos() -> mb_strrpos()
5. substr() -> mb_substr()
6. strtolower() -> mb_strtolower()
7. strtoupper() -> mb_strtoupper()
8. substr_count() -> mb_substr_count()
9. ereg() -> mb_ereg()
10. eregi() -> mb_eregi()
11. ereg_replace() -> mb_ereg_replace()
12. eregi_replace() -> mb_eregi_replace()
```

13. split() -> mb_split()

Alles anzeigen

Die Funktion htmlentities ignoriert den Zeichensatz per Default auch. Daher muss man ihr den dritten Funktionsparameter immer mitgeben:

Quellcode

1. htmlentities($var, ENT_QUOTES, 'UTF-8')

== Content Type ==
Dateien haben eine Zeichenkodierung und Editoren speichern in dieser. Arbeiten mehrere Personen mit unterschiedlichen Zeichensätzen an der selben Datei kann dies fatale Auswirkungen haben. Daher sollte jeder seine Entwicklungsumgebung auf UTF-8 umstellen.

Bei HTTP Requests sollte dazu der Zeichensatz auch übertragen werden. Benutzt dazu das header Statement.

Quellcode

1. header('Content-type: text/html; charset=UTF-8') ;

Damit der Client weiß welcher Zeichensatz ihn erwartet darf man auch die Meta-Angabe nicht vergessen:

Quellcode

1. <meta http-equiv="Content-type" value="text/html; charset=UTF-8" />

== Datenbank Connection Encoding ==
Bei der Nutzung von PHP sollte man das Connection Encoding auf UTF-8 erzwingen.

Quellcode

1. $dbh = new PDO('mysql:...');
2. $dbh->exec("SET NAMES utf8");

== Literatur ==

- nicknettleton.com/zine/php/php-utf-8-cheatsheet