

Seltsame Zeichen

“Warum werden auf meiner Seite Umlaute und Sonderzeichen falsch dargestellt?”

Warum?

Nicht selten gibt es Themen auf [php.de](https://www.php.de) wie diese. Probleme mit Umlauten und anderen Sonderzeichen, die fehlerhaft dargestellt werden. In folgenden werden diese Situationen mit Hintergründen und Lösungen dazu, basierend auf den beiden Zeichenkodierungen UTF-8 (Unicode) und ISO 8859-1 (Latin-1), gezeigt.

Situation 1

Wenn ein Dokument als Unicode (UTF-8) abgespeichert wurde, aber als ISO 8859-1 interpretiert wird, geschieht folgendes:

```
Erwartete Ausgabe: ä ö ü  
Wirkliche Ausgabe: Ã ¤ Ã ¶ Ã ¼
```

Das ä wurde binär gesehen als 11000011 10100100 gespeichert. Diese beiden Bytes gehören laut UTF-8 zusammen, werden in ISO 8859-1 allerdings auseinander genommen. Folglich wird aus 11000011 ein Ã und aus 10100100 ein ¤-Zeichen.

Was zu beachten ist, damit es zu keiner solchen Fehlinterpretation kommt, ist [etwas weiter unten](#) angeführt.

Situation 2

Wenn die Umlaute als Fragezeichen dargestellt werden, dann wird ein als ISO 8859-1 kodierte Dokument als UTF-8 interpretiert. Dies stellt das Gegenstück zur oben genannten Situation dar.

```
Erwartete Ausgabe: ä ö ü  
Wirkliche Ausgabe: ? ? ?
```

Mit anderen Worten wurde hier ein ä als 11100100 gespeichert. In UTF-8 ist dieses Byte jedoch **ungültig**, wird also nicht angezeigt (ignoriert) bzw. als ? oder ? dargestellt.

Was zu beachten ist, damit es zu keiner solchen Fehlinterpretation kommt, ist [nachfolgend](#) angeführt.

Lösung: Nutze konsequent UTF-8

Checkliste für die durchgängige und korrekte Verwendung von UTF-8

- PHP- und HTML-/Template-Dateien im Editor als “UTF-8 ohne BOM” speichern. Dies betrifft auch alle etwaigen serverseitigen inkludierten Dateien.

- HTTP Header Content-Type mit UTF-8 verwenden. `header( 'Content-Type: text/html; charset=UTF-8' );`. Der HTTP-Header hat (abgesehen vom BOM) die höchste Priorität bei der Bestimmung der Zeichenkodierung.
- HTML-meta-Tag
HTML 4.x: `<meta http-equiv=„content-type“ content=„text/html; charset=utf-8“ />`
HTML 5: `<meta charset=„utf-8“>`
- Formulardaten (falls explizit nötig) in UTF-8 übergeben (ggf. mit `accept-charset=„utf-8“` sicherstellen)
- Datenbankverbindung von PHP zur Datenbank auf UTF-8 stellen. Siehe z.B. [hier für PDO](#). Für `mysqli` gibt es die Methode `mysqli_set_charset()`. Siehe dazu auch [MySQL und UTF-8](#)
- Datenbank Zeichensatz UTF-8, Tabellenkollationen `utf8_unicode_ci`. Siehe dazu auch [MySQL und UTF-8](#)
- Daten aus Fremddaten müssen mittels `utf8_encode()` in UTF-8 überführt werden, wenn sie nicht als UTF-8 vorliegen (Datei, Windows-System, ...).
- Bei der Anwendung von `htmlspecialchars()` und `htmlentities()` die Codierung mitgeben. Beispiel (HTML 5) für `htmlspecialchars()`:

```
function escape($s) {  
    return htmlspecialchars(  
        $s,  
        ENT_QUOTES | ENT_HTML5 | ENT_DISALLOWED | ENT_SUBSTITUTE,  
        'UTF-8'  
    );  
}
```

- Generell ist darauf zu achten, dass es zu keinen "Überkodierungen" kommt, zB wenn ein bereits als UTF-8 kodierter String erneut mittels `utf8_encode()` behandelt wird.

Weiterführende Links / Quellen

- [Angabe der Zeichencodierung in HTML \(W3C\)](#)
- [Zeichencodierung für Anfänger \(W3C\)](#)
- [Auflistung gesammelter Punkte von jspit \(php.de\)](#)
- [Umlautprobleme \(floern.com\)](#)

From:
<https://wiki.hennweb.de/> - **HennWeb**

Permanent link:
<https://wiki.hennweb.de/doku.php?id=programmieren:allgemein:umlaute&rev=1610111591>

Last update: **08/01/2021 14:13**

