

UTF-8 web-applications using Dojo&PHP

v0.2, Matthias Kerstner, <http://www.kerstner.at>

This document presents you a list of things to keep in mind when creating UTF-8 aware web-applications using Dojo and PHP. Although the focus lies on Dojo and PHP most of the tips below can be used for any other scripting language too.

- **write files UTF8-encoded with an UTF8-capable editor, like [PSPad](#)**

- **(optional) insert the following in your .htaccess**

```
php_value default_charset UTF-8
```

This will direct PHP to always send out data UTF-8 encoded.

- **specify correct character set in HTML header**

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
```

- **specify correct language in HTML header**

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
```

- **force Dojo to use UTF-8 for dojo.iobind**

```
var djConfig = {isDebug:true, parseOnLoad:false, bindEncoding:"UTF-8"};
```

- **specify character set for your <form>**

```
<form accept-charset="UTF-8">
```

Although this is optional (browsers will automatically use the encoding previously used by the server) when specified it is much more obvious what encoding you want to use.

- **define character set to be used for returning data on the server side**

```
header("Content-Type: text/html; charset=utf-8");
```

- **encode data to be sent to the server on the client side**

```
encodeURIComponent()
```

This makes sure that special characters such as "\$%&,..." will be transferred correctly

- **decode received data on the server side**

```
html_entity_decode(urldecode($p), ENT_QUOTES, "UTF-8")
```

- **use `utf8_general_ci` as collation for your MySQL database**

- **right before reading/writing from your database issue the following query**

```
mysql_query("SET NAMES 'utf8'")
```

This will direct the MySQL server to await UTF-8 encoded data

- **escape quotation marks to avoid code-injection when issuing database queries**

```
addSlashes()
```

- **when using PHP's json_encode() function be sure to utf8_encode strings as json_encode will cut off german umlauts that were not encoded before**

```
json_encode(utf8_encode(rawurlencode($x))
```

Summary:

Client: `encodeURIComponent()` , `decodeURIComponent`

Server: `html_entity_decode(urldecode($p), ENT_QUOTES, "UTF-8")`

```
mysql_query("SET NAMES 'utf8'"), addSlashes()
```

```
header("Content-Type: text/html; charset=utf-8")
```

```
json_encode(utf8_encode(rawurlencode($x))
```

MySQL: `utf8_general_ci`

Using this technique your database will contain the original data entered by the user (without any extra encoding, like by `htmlentities()`). Additionally you should be safe from SQL-code injection by using `addSlashes()`. If you have anything to add please contact me through the contact form:

<http://kerstner.at/?id=3>

Matthias Kerstner

31.01.2008