eZ systems

Share your information

Welcome!

Multi-language Development

Intl. PHP Conference

November 9th, 2004. Frankurt, Germany

Derick Rethans <derick@php.net>

http://pres.derickrethans.nl/wereldveroverend-ffm2004

eZ systems

Not Everything is ASCII

- ASCII is a 7 bit character set
- ASCII is old (published in 1968)
- Only English does not require accented characters.
- There are more languages than English :)
- This means that it is no longer adequate

	2	3	4	5	6	7
0:		0	@	P	`	p
1:	!	1	A	Q	a	q
2:	"	2	B	R	b	r
3:	#	3	C	S	c	s
4:	$	4	D	T	d	t
5:	%	5	E	U	e	u
6:	&	6	F	V	f	v
7:	'	7	G	W	g	w
8:	(	8	H	X	h	x
9:	)	9	I	Y	i	y
A:	*	:	J	Z	j	z
B:	+	;	K	[	k	{
C:	,	<	L	\	l	|
D:	-	=	M	]	m	}
E:	.	>	N	^	n	~
F:	/	?	O	_	o	DEL

Share your information

eZ systems

ISO-8859 and Latin Character Sets

- ISO-8859: Group of often used character sets
- latin-X: Group of character sets based on the "latin" set
- Contains most of European's characters
- Adds 96 characters to ASCII (A0 - FF)

ISO 8859	Latin	Usage	Example
1	1	West European languages	é ë ç å
2	2	Central and East European languages	š Č ř
3	3	Southeast European and miscellaneous languages	
4	4	Scandinavian/Baltic languages	
5		Latin/Cyrillic	б ж п д
6		Latin/Arabic	ت ج ق
7		Latin/Greek	α ζ π ώ
8		Latin/Hebrew	ם צ ט א
9	5	Latin-1 modification for Turkish	Ş ş İ ÿ
10	6	Lappish/Nordic/Eskimo languages	
11		Latin/Thai	ก ณ ท ฟ
13	7	Baltic Rim languages	
14	8	Celtic	
15	9	West European languages	€
16	10	Romanian	Đ Ę Ţ đ

Share your information

eZ systems

Conversion Between ISO-8859 Character Sets

- Each set has only 256 positions
- Impossible to convert everything
- Conversion will result in broken text
- http://www.eki.ee/letter/

ISO 8859-1		ISO-8859-2		
Hex	Char	Hex	Char	Description
E0	à	--	-	LATIN SMALL LETTER A WITH GRAVE
E1	á	E1	á	LATIN SMALL LETTER A WITH ACUTE
E2	â	E2	â	LATIN SMALL LETTER A WITH CIRCUMFLEX
E3	ã	--	-	LATIN SMALL LETTER A WITH TILDE
E4	ä	E4	ä	LATIN SMALL LETTER A WITH DIAERESIS
E5	å	--	-	LATIN SMALL LETTER A WITH RING ABOVE
E6	æ	--	-	LATIN SMALL LETTER AE
E7	ç	E7	ç	LATIN SMALL LETTER C WITH CEDILLA
E8	è	--	-	LATIN SMALL LETTER E WITH GRAVE
E9	é	E9	é	LATIN SMALL LETTER E WITH ACUTE
EA	ê	--	-	LATIN SMALL LETTER E WITH CIRCUMFLEX
EB	ë	EB	ë	LATIN SMALL LETTER E WITH DIAERESIS
EC	ì	--	-	LATIN SMALL LETTER I WITH GRAVE
ED	í	ED	í	LATIN SMALL LETTER I WITH ACUTE
EE	î	EE	î	LATIN SMALL LETTER I WITH CIRCUMFLEX
EF	ï	--	-	LATIN SMALL LETTER I WITH DIAERESIS

Share your information

Asian Character Sets

亚字

- They need more than 256 positions
- Multi-byte
- One character is often one word
- Problems with NULL characters and special characters
- addslashes() mis-recognises chinese-big5 words like "0xA5 0x5C" (¥\, 功, E5 8A 9F)

倪存内洋国

Unicode and ISO-10646 (UCS)

- UCS uses 31 bits for character storage
- Contains all known characters and symbols
- First 128 bytes are the same as ASCII
- First 256 bytes are the same as ISO-8859-1
- Unicode 3.0 describes the BMP (16 bits)
- Unicode 3.1 describes other planes (21 bits)
- Characters are ordered in language/script blocks: Basic latin, Cyrillic, Hebrew, Arabic, Gujarati, Runic, CJK etc.
- Encoding in numerous encodings: UCS-2, UCS-4, UTF8, UTF16 etc.

A Њ ث א અ ᛋ 媛

UTF8

Encoding of UCS characters

- All UCS characters > 0x7f (127) are stored as multi-byte characters >= 0x80
- No problems with control characters
- The first byte of a multi-byte character is always in the range 0xc0 - 0xfd and describes how long this byte-sequence is.
- European languages usually use up to two bytes per character
- Characters of other languages usually use up to three bytes per character
- The longest UTF8 encoded character can be 6 bytes, but only 4 bytes are used

UTF8 Encoding

Characters in the ASCII range are stored "as-is"

Characters in the range 0x0080 to 0x07ff:

```
byte 1: 110x xxxx
byte 2: 10xx xxxx

ë (0x00eb)   110x xxxx  10xx xxxx
1110 1011 -> ---+ ++11  --10 1011
             ====================
             1100 0011  1010 1011
                  0xc3       0xab
```

Characters in the range 0x8000 to 0xffff:

```
byte 1: 1110 xxxx
byte 2: 10xx xxxx
byte 3: 10xx xxxx

功 (0x529f)              1110 xxxx  10xx xxxx  10xx xxxx
0101 0010  1001 1111 ->  ---+ 0101  --00 1010  --01 1111
                         ==============================
                         1110 0101  1000 1010  1001 1111
                              0xe5       0x8a       0x9f
```

eZ systems

Problems with Multi-byte Characters

strlen() no longer "works":

```php
<?php
$str = "Vær så god!";
echo "The string has length: ", strlen($str);
?>
```

output:

```
The string has length: 13
```

substr() may mangle text:

```php
<?php
$str = "Vær så god!";
echo "The first 7 characters are: ", substr($str, 0, 7);
?>
```

output:

```
The first 7 characters are: Vær s
```

wordwrap() wraps too early:

```php
<?php
$str = "Vær så god!";
echo wordwrap($str, 6, '|');
?>
```

output:

```
Vær|så|god!
```

Share your information

Iconv Functions

iconv_strlen():

```php
<?php
$str = "Vær så god!";
echo "The string has length: ", iconv_strlen($str, 'utf8');
?>
```

output:

```
The string has length: 11
```

iconv_substr():

```php
<?php
iconv_set_encoding('internal_encoding', 'utf8');
$str = "Vær så god!";
echo "The first 7 characters are: ", iconv_substr($str, 0, 7);
?>
```

output:

```
The first 7 characters are: Vær så
```

eZ systems

Share your information

Iconv Functions

iconv_strpos() and iconv_strrpos()

iconv_strpos():

```php
<?php
iconv_set_encoding('internal_encoding', 'utf8');
$str = "Vær så god!";
echo "The o is at position: ", strpos($str, 'o'), "<br />\n";
echo "The o is at position: ", iconv_strpos($str, 'o'), "\n";
?>
```

output:

```
The o is at position: 10
The o is at position: 8
```

iconv_strrpos():

```php
<?php
iconv_set_encoding('internal_encoding', 'utf8');
$str = "Vær så god!";
echo "The å is at position: ", strrpos($str, 'å'), "<br />\n";
echo "The å is at position: ", iconv_strrpos($str, 'å'), "\n";
?>
```

output:

```
The å is at position: 6
The å is at position: 5
```

Regular Expressions and UTF8

preg_match():

```php
<?php
$str = "1978,Dérick,Rethans";
if (preg_match('@,(.{6}),@', $str, $result)) {
    echo 'We matched: ', $result[1], "<br/>\n";
} else {
    echo 'There was no match!', "<br/>\n";
}
?>
```

output:

```
There was no match!
```

preg_match() in UTF8 mode:

```php
<?php
$str = "1978,Dérick,Rethans";
if (preg_match('@,(.{6}),@u', $str, $result)) {
    echo 'We matched: ', $result[1], "<br/>\n";
} else {
    echo 'There was no match!', "<br/>\n";
}
?>
```

output:

```
We matched: Dérick
```

eZ systems

Share your information

Iconv

Character Set Conversions

- Is part of glibc
- BSD requires an external library
- Enabled by default in PHP 5
- Supports 100s of character sets:

DOS, ISO-8859, MAC, EBCDIC, UTF*, IBM, OSF...

Converting Character Sets

iconv():

```php
<?php
$str = "1978,Dérick,Rethans";
echo iconv('iso-8859-1', 'utf8', $str);
?>
```

output:

```
1978,DÃ©rick,Rethans
```

iconv():

```php
<?php
$str = "© 2004, by Derick Rethans";
echo iconv('iso-8859-1', 'iso-8859-2', $str);
?>
```

output:

```
Notice: iconv() [function.iconv]: Detected an illegal character in input string in
/home/httpd/pres2/display.php(461) : eval()'d code on line 3
```

iconv():

```php
<?php
$str = "© 2004, by Derick Rethans";
echo iconv('iso-8859-1', 'iso-8859-2//ignore', $str);
?>
```

output:

```
Notice: iconv() [function.iconv]: Detected an illegal character in input string in
/home/httpd/pres2/display.php(461) : eval()'d code on line 3
  2004, by Derick Rethans
```

eZ systems

Automatic Conversions

Iconv and Output Buffers

Share your information

ob_iconv_handler():

```php
<?php
    // Output text as-is
    echo 'Nittenhundreogåttiåtte', "<br/>\n";

    // Set the internal and output encodings
    iconv_set_encoding("internal_encoding", "iso-8859-1");
    iconv_set_encoding("output_encoding", "utf-8");

    // Start the output buffer handler and echo the string again
    ob_start("ob_iconv_handler");
    echo 'Nittenhundreogåttiåtte', "<br/>\n";

    // Flush and stop the output buffer handler
    ob_end_flush();
?>
```

output:

```
Nittenhundreogåttiåtte
NittenhundreogÃ¥ttiÃ¥tte
```

Conversions with Multiple Character Sets

ob_iconv_handler():

```php
<?php
    // Output text as-is
    echo 'Nittenhundreogåttiåtte', "<br/>\n";

    // Set the output and internal encodings
    iconv_set_encoding("output_encoding", "utf-8");
    iconv_set_encoding("internal_encoding", "iso-8859-1");

    // Start the output buffer handler and echo the string again
    ob_start("ob_iconv_handler");
    echo 'Nittenhundreogåttiåtte', "<br/>\n";
    ob_end_flush();

    // Change the internal encoding
    iconv_set_encoding("internal_encoding", "iso-8859-2");

    // Start the output buffer handler and echo the string again
    ob_start("ob_iconv_handler");
    echo 'Nittenhundreogåttiåtte', "<br/>\n";
    ob_end_flush();
?>
```

output:

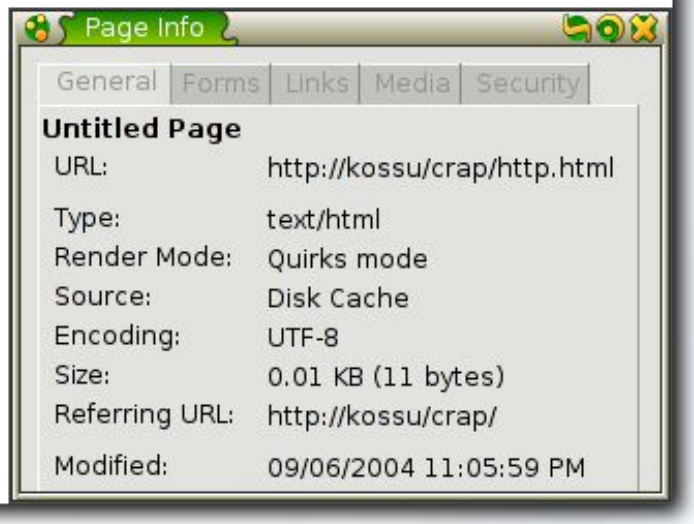
```
Nittenhundreogåttiåtte
NittenhundreogÃ¥ttiÃ¥tte
NittenhundreogÄªttiÄªtte
```

HTTP and Browser Problems

- Most sites don't set any Content-Type header
- Browsers and/or digital environments will move more and more to UTF-8 as default encoding

Result:

V&#65533;r s&#65533; god



eZ systems

Setting a Content-Type header

```
<?php
    header("Content-Type: text/html; charset=iso-8859-1");
?>
Vær så god
```

or php.ini setting:

```
default_charset = "utf-8"
```

Very easy to do, and the result is much better:

Share your information

eZ systems

Conversion to HTML

(slide encoding is iso-8859-1)

iso-8859-1 to HTML:

```php
<?php
    $str = "Nittenhundreogåttiåtte<br/>\n";
    echo htmlentities($str, ENT_NOQUOTES, 'iso-8859-1');
?>
```

output:

```
Nittenhundreog&aring;tti&aring;tte&lt;br/&gt;
```

UTF-8 to HTML:

```php
<?php
    $str = "NittenhundreogÃ¥ttiÃ¥tte<br/>\n";
    echo htmlentities($str, ENT_NOQUOTES, 'utf-8');
?>
```

output:

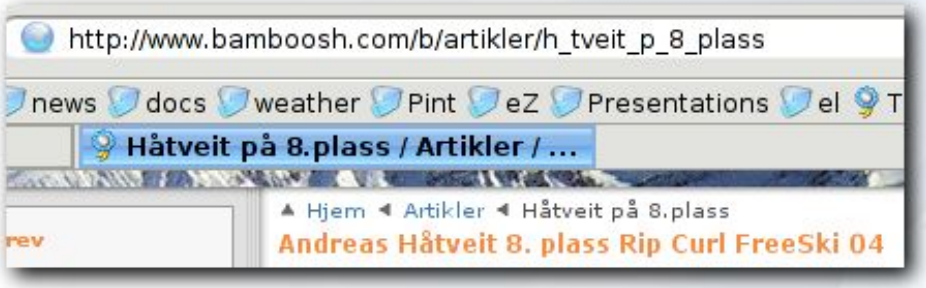
```
Nittenhundreog&aring;tti&aring;tte&lt;br/&gt;
```

Share your information

eZ systems

Transliteration

"Funny Characters"



What we do now:

```
"Håtveit på 8. plass" to "h_tveit_p_8_plass"
```

What we want to do:

```
"Håtveit på 8. plass" to "haatveit_paa_8_plass"
```

How we do this:

```php
<?php
$string = iconv("latin1", "ucs-2", "Håtveit på 8. plass");
$res = transliterate($string,
    array('normalize_ligature', 'lowercase_latin', 'space_to_underscores'));
echo iconv('ucs-2', 'latin1', $res);
?>
```

Share your information

Transliteration

Cyrillic Characters

```php
<?php
    $string = <<<END
След малко се запътвам със автобусната спирка, от там на
летището, после пак на летището и пак на автобусната спирка
и в Пловдив.
END;

    $string = iconv("utf-8", "ucs-2", $string);
    $res = transliterate($string, array('cyrillic_transliterate'));
    echo iconv('ucs-2', 'utf-8', $res);
?>
```

output:

```
Sled malko se zap'tvam s'm avtobusnata spirka, ot tam na letischeto, posle pak na
letischeto i pak na avtobusnata spirka i v Plovdiv.
```

eZ systems

Transliteration

Chinese, Hebrew and Greek Characters

```php
<?php
    $string = <<<END
美军总攻费卢杰 战况惨烈

מורדים בפלוג'ה משגרים רקטה בניסיון לעצור את הניסיון של כוחות
ארה"ב לכבוש את העיר תצלום: אי-פי

Υποθέτω πως για τους ενασχολούντες, η είδηση της βάφτισης
του linux kernel tree σε 2.6 στις 17 του Δεκέμβρη 2003 είναι
ήδη γνωστή.
END;

    $string = iconv("utf-8", "ucs-2", $string);
    $res = transliterate($string, array(
        'han_transliterate', 'hebrew_transliterate', 'greek_transliterate'));
    echo nl2br(iconv('ucs-2', 'utf-8', $res));
?>
```

output:

```
měijūnzǒnggōngfèilújié zhànkuàngcǎnliè

mordym bplog'h msgrym rqth bnysyon l'zor 't hnysyon sl kohot
'rh"b lkbos 't h'yr tzlom: 'y-py

Ypothheto pos gia toys enascholohuntes, e ehidese tes vhaphtises
toy linux kernel tree se 2.6 stis 17 toy Dekhemvre 2003 ehinai
hede gnosthe.
```

Share your information

Normalization

```php
<?php
    $string = <<<END
Normalization:
This example is called «normalization». It`s used to convert of
“curly quotes”, special—hypens and other characters to more ‘ascii’
representations.

Removing diacritical marks:
Ḁńǿţḫệŕ ḟíḻţęṛ ŕéḿṓṽȩṣ ḍįąçŕịṫıçáḹ ṁẵŕķṩ.
END;

    $string = iconv("utf-8", "ucs-2", $string);
    $res = transliterate($string, array(
        'normalize_punctuation', 'diacritical_remove'));
    echo nl2br(iconv('ucs-2', 'utf-8', $res));
?>
```

output:

```
Normalization:
This example is called "normalization". It's used to convert of
"curly quotes", special--hypens and other characters to more 'ascii'
representations.

Removing diacritical marks:
Another filter removes diacritical marks.
```

Other filters

```
<pre><?php
    $string = <<<END
decompose_currency_signs: € £ ¥
decompose_special:        © ± « »
normalize_numbers:        ६.७५ * ൫൮൨൩.६५३ + ㈥㈡
han_transliterate:        乢乡乞乃
uppercase_cyrillic:       След малко се запътвам
END;

    $string = iconv("utf-8", "ucs-2", $string);
    $res = transliterate($string, array(
        'decompose_currency_signs', 'decompose_special', 'normalize_numbers',
        'han_transliterate', 'uppercase_cyrillic', 'normalize_ligature',
        'diacritical_remove'));
    echo iconv('ucs-2', 'utf-8', $res);
?>
```

output:

```
decompose_currency_signs: eur L Y
decompose_special:        (c) +- << >>
normalize_numbers:        6.75 * 5823.653 + 62
han_transliterate:        gaixiangqinai
uppercase_cyrillic:       СЛЕД МАЛКО СЕ ЗАПЪТВАМ
```

Locales

language and cultural rules

A locale is a set of language and cultural rules. These cover aspects such as language for messages, different character sets, lexigraphic conventions, etc.

Locale types:

- LC_COLLATE: string collation
- LC_CTYPE: character properties, case sensivity
- LC_MONETARY: monetary formatting
- LC_NUMBER: number formatting
- LC_TIME: date and time formatting

eZ systems

Share your information

LC_COLLATE

language oriented sorting

```php
<?php
    $b = $a = array("øks", "ærlig", "åtte", "øyeblikk");

    setlocale(LC_COLLATE, 'C');
    sort($a);
    var_dump($a);

    setlocale(LC_COLLATE, 'no_NO.UTF8');
    sort($b, SORT_LOCALE_STRING);
    var_dump($b);
?>
```

output:

```
array
  0 => 'åtte'
  1 => 'ærlig'
  2 => 'øks'
  3 => 'øyeblikk'

array
  0 => 'ærlig'
  1 => 'øks'
  2 => 'øyeblikk'
  3 => 'åtte'
```

Mathematical order: å (229), æ (230), ø (248)
Natural order: æ, ø, å

eZ systems

LC_CTYPE

language oriented case changes

```php
<?php
    $a = "øyeblikk";

    setlocale(LC_CTYPE, 'C');
    echo strtoupper($a), "<br/>\n";

    setlocale(LC_CTYPE, 'no_NO');
    echo strtoupper($a), "<br/>\n";

    setlocale(LC_CTYPE, 'tr_TR');
    echo strtoupper($a), "<br/>\n";
?>
```

output:

```
øYEBLIKK
ØYEBLIKK
ØYEBLÝKK
```

Share your information

eZ systems

LC_TIME

language oriented date and time formatting

```
<pre><?php
    $locales = array(
        'Arabic (Egypt)' => 'ar_EG.UTF-8', 'American' => 'en_US',
        'Dutch   '       => 'nl_NL',       'Hebrew  ' => 'iw_IL',
        'Hebrew (UTF-8)' => 'iw_IL.UTF-8', 'Japanese' => 'ja_JP.UTF-8',
        'Norwegian'      => 'no_NO.UTF-8', 'Turkish ' => 'tr_TR.UTF-8'
    );

    foreach ($locales as $country => $locale) {
        setlocale(LC_TIME, $locale);
        echo '<b>', $country, "</b>\t", strftime('%c'),
            "\n";
    }
?></pre>
```

output:

```
Arabic (Egypt)  09 2004 ,نوف CET 02:29:08 م
American        Tue 09 Nov 2004 02:29:08 PM CET
Dutch           di 09 nov 2004 14:29:08 CET
Hebrew          CET 14:29:08 2004 ◆◆◆ 09 ◆'
Hebrew (UTF-8)  CET 14:29:08 2004 ג 09 נוב'
Japanese        2004年11月09日 14時29分08秒
Norwegian       tir 09-11-2004 14:29:08 CET
Turkish         Sal 09 Kas 2004 14:29:08 CET
```

Share your information

eZ systems

LC_NUMERIC

formatting numbers

```
<pre><?php
    $locales = array(
        'Arabic (Egypt)' => 'ar_EG.UTF-8', 'American' => 'en_US',
        'Dutch   '       => 'nl_NL',       'Hebrew  ' => 'iw_IL',
        'Hebrew (UTF-8)' => 'iw_IL.UTF-8', 'Japanese' => 'ja_JP.UTF-8',
        'Norwegian'      => 'no_NO.UTF-8', 'Turkish ' => 'tr_TR.UTF-8'
    );

    foreach ($locales as $country => $locale) {
        setlocale(LC_NUMERIC, $locale);
        $ldata = localeconv();
        echo "<b>$country</b>\t",
             number_format("31415.92654", 2, $ldata['decimal_point'],
             $ldata['thousands_sep']), "\n";
    }
?></pre>
```

output:

```
Arabic (Egypt)  31,415.93
American        31,415.93
Dutch           31415,93
Hebrew          31,415.93
Hebrew (UTF-8)  31,415.93
Japanese        31,415.93
Norwegian       31.415,93
Turkish         31415.93
```

Share your information

eZ systems

LC_MONETARY

formatting amounts of money

```
<pre><?php
$locales = array(
  'Egypt   ' => 'ar_EG.UTF-8', 'Finland  ' => 'fi_FI.UTF-8',
  'Germany ' => 'de_DE.UTF-8', 'Israel   ' => 'iw_IL.UTF-8',
  'Japan   ' => 'ja_JP.UTF-8', 'Netherlands' => 'nl_NL.UTF-8',
  'Norway  ' => 'no_NO.UTF-8',
);

foreach ($locales as $country => $locale) {
  setlocale(LC_MONETARY, $locale);
  $nr = 31415.92654;
  echo "<b>$country</b>\t",
    money_format("[%i]", $nr), "    \t", money_format("[%n]", $nr), "\n";
}
?></pre>
```

output:

```
Egypt          [EGP 31,415.927]        [31,415.927 .ج.م]
Finland        [31 415,93EUR]          [31 415,93€]
Germany        [31.415,93 EUR]         [31.415,93 €]
Israel         [ILS 31,415.93]         [31,415.93 ₪]
Japan          [JPY 31,416]            [￥31,416]
Netherlands    [EUR 31 415,93]         [€ 31 415,93]
Norway         [NOK31.415,93]          [kr31.415,93]
```

Share your information

Locales and PHP

string **setlocale**(const *category*, string *locale*, ...)
string **setlocale**(const *category*, array *locales*)

Localenames:

- Are OS dependent (deu_deu vs. de_DE)
- Are not always available

Example:

```php
<?php
    $locale = setlocale(LC_ALL,
        "nl_NL.UTF-8@euro", "nl_NL.UTF-8", "dutch", "nld_nld");
    echo "New locale: ", $locale, "<br />\n";

    $locale = setlocale(LC_ALL, array("nld_nld", "nederlands"));
    echo "New locale: ", $locale, "<br />\n";
?>
```

output:

```
New locale: nl_NL.UTF-8@euro
New locale:
```

Resources

multi-language development

This presentation:
http://pres.derickrethans.nl/wereldveroverend-phpworks04

Questions?: derick@php.net