Fourth Annual

MySQL®

Users Conference 2006

Practical I18N with PHP and MySQL

Jim Winstead
MySQL Inc.
jimw@mysql.com

MySQL Users Conference 2006
April 25, 2006

Presented by MySQL O'REILLY

What is I18N?

- Internationalization (I18N)
  - "The practice of designing and writing programs that can be easily configured to act with the user in more than one language."
- Localization (L10N)
  - "The development process that customizes software and documentation for use in a specific country or language environment."

What do I have to worry about?

- What are you sending the browser?
- What is the browser sending you?
- What are you storing in the database?
- How do you sort strings?

Character Sets and Character Encodings

- http://www.w3.org/TR/charmod/
  - “A ‘character encoding’ is a mapping from a character set definition to the actual code units used to represent the data.”

Unicode

- **http://www.unicode.org/**
  http://www.unicode.org/faq/unicode_web.html

0400 **Cyrillic**

	040	041	042	043	044	045	046	047	048	049
0	È 0400	А 0410	Р 0420	а 0430	р 0440	è 0450	Ѡ 0460	Ψ 0470	Ҫ 0480	Г 0490
1	Ё 0401	Б 0411	С 0421	б 0431	с 0441	ё 0451	ѡ 0461	ψ 0471	ҫ 0481	г 0491
2	Ђ 0402	В 0412	Т 0422	в 0432	т 0442	ђ 0452	Ѣ 0462	Ѳ 0472	≠ 0482	Ғ 0492

UTF-8

Table 3-6. Well-Formed UTF-8 Byte Sequences

Code Points	1st Byte	2nd Byte	3rd Byte	4th Byte
U+0000..U+007F	00..7F			
U+0080..U+07FF	C2..DF	80..BF		
U+0800..U+0FFF	E0	***A0***..BF	80..BF	
U+1000..U+CFFF	E1..EC	80..BF	80..BF	
U+D000..U+D7FF	ED	80..***9F***	80..BF	
U+E000..U+FFFF	EE..EF	80..BF	80..BF	
U+10000..U+3FFFF	F0	***90***..BF	80..BF	80..BF
U+40000..U+FFFFF	F1..F3	80..BF	80..BF	80..BF
U+100000..U+10FFFF	F4	80..***8F***	80..BF	80..BF

Collation

- “The process of ordering units of textual information.”
- This depends on the language of the content, which you may or may not be able to determine based on the character set.
- The Unicode Collation Algorithm

What should you send?

- What do you have?
- What do you want back?
- Set in the header:

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

- And in the document:

```
<?xml version="1.0" encoding="utf-8"?>
<meta http-equiv="Content-type" content="text/html;
charset=utf-8" />
```

What is usually sent?

- The default encoding for HTML is ISO-8859-1
- The default encoding for XML is UTF-8
- …but that depends on the Content-type. (See RFC 3023.)

Handling funny characters in HTML

- å = å = å = å
- This is HTML. å may not equal å in all XML (or SGML) documents.
- Å = Å = &#C5; = Å
  except:
  Å = Å = Å

What will you get back?

- Browsers generally don't tell you what character set they've sent.
- If you send UTF-8, you should get back UTF-8.
- If you send another character set, you should get back that character set.
- Test for valid UTF-8: `http://xrl.us/kyi9`

Using a hidden field

- **<input type="hidden" name="charset-check" value="Φ♭ǀB" />**
- Check what you got in the hidden field, which should help you determine what you got in the other fields.

What are you storing in the database?

- If you can constrain yourself to one charset, use that! (But don't forget to code your application to scream when it gets stuff it can't handle.)
- The most defensive choice is UTF-8.

Converting the data

- mbstring extension:
  `mb_convert_encoding(string, to, from)`
- iconv extension:
  `iconv(from, to, string)`
- recode_extension:
  `recode_string(request, string)`
- `utf8_encode()` (but it only handles iso-8859-1 to utf-8)

Dealing with XML from PHP

- PHP 4 does not natively handle XML in arbitrary input encodings.
- It is hardcoded to default to ISO-8859-1, and you can only tell it to use a different encoding, not fall back to the encoding detection as defined in the XML standard.

But PHP 5 is better

- PHP 5 improves this situation slightly: it still defaults to ISO-8859-1, but you can tell it to behave the right way.
- A way to work around things from Steve Minutillo:
  `http://xrl.us/kyie`

Character Sets in 4.1

- Per-table, database, and server
- Default character set is latin1 (Windows 1252) and default collation is latin1_swedish_ci

Introducers

- `_latin1"fiancée" COLLATE latin1_general_cs`
- Does not convert the string, but tells the server what charset that string literal is in

Converting

- `CONVERT(field USING utf8)`
- `CAST(field AS VARCHAR(255) CHARACTER SET utf8)`

Using collations

- Easy: `SELECT x FROM T ORDER BY x;`
- Tricky: `SELECT x FROM T WHERE x = 'Y';`

Collation coercibility

- An explicit COLLATE clause has a coercibility of 0. (Not coercible at all.)
- The concatenation of two strings with different collations has a coercibility of 1.
- The collation of a column or a stored routine parameter or local variable has a coercibility of 2.
- A “system constant” (the string returned by functions such as USER() or VERSION()) has a coercibility of 3.
- A literal's collation has a coercibility of 4.
- NULL or an expression that is derived from NULL has a coercibility of 5.

Converting from 4.0

- Not a problem — unless you cheated
- You can use the BINARY field type to transition:
  - On 4.0:

```
ALTER TABLE t1 MODIFY utf8_col BINARY(255);
```

  - On 4.1:

```
ALTER TABLE t1 MODIFY utf8_col CHARACTER
SET utf8;
```

Some notes on L10N

- gettext extension: keep your application in English, but put text strings in gettext() (or _()) function calls
- Not a great solution for a text-heavy web application or site.
- Template systems like Smarty have solutions for this

The future

- PHP 6 will have native Unicode string handling. It's a work-in-progress.

Fourth Annual

MySQL®

Users Conference 2006

Practical I18N with PHP and MySQL

Jim Winstead
MySQL Inc.
jimw@mysql.com

MySQL Users Conference 2006
April 25, 2006

Presented by MySQL O'REILLY