The Mysteries of Charsets
ä 迎 û

T3DD09 - Mai 2009 - HH-Elmshorn

Peter Niederlag

TYPO3

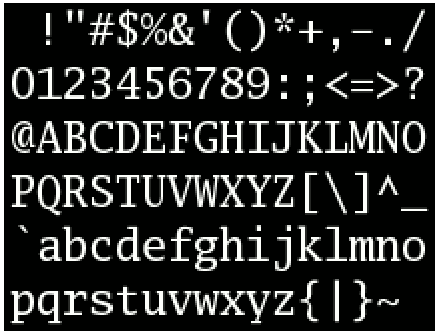
TOC

- Computers know 0 and 1
- Example
  - Take 4 Coins and Toss
  - 16 different possibilities
- binary represantation
  - 0000 - 1111
- human represantation:
  - 16 numbers?
  - 16 letters?
  - 16 football teams?

- Humans use Characters not Bytes in interaction
- Computers transform Characters to Bytes
- The transformation needs a standard/rule
  – C => 0001001 ? 0101010 ?
- The Computer needs to know which standard/rule to use
- ASCII as first standard/rule, or „encoding“

ASCII

- ~ 1960
- 7 bits => 128 pieces
  - 95 printable characters
  - 33 control characters (line feed, return, tabulator, ...)
- In the „Beginning“ the nerds didn't think of anything but a-z, A-Z, 0-9 and some special characters



```
 !"#$%&'()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmno
pqrstuvwxyz{|}~
```

ASCII printable characters

Peter Niederlag

Example ASCII

- Examples (ASCII)

character/glyph	binary	decimal	hex
Z	101 1010	90	5A
{	111 1011	123	7B
ü	--	--	--
€	--	--	--

IS0-8859-1, latin1, ISO-8859-15 and cp1252

- ISO-8859-1,latin1, ISO-8859-15 and cp1252
- These use 8 bits / 1 Byte for encoding
- Basically they are all the same ;)
  - ISO-8859-1 and latin1 ARE the same
  - ISO-8859-15 is an adoption with €
  - cp1252 (windows) is an adoption with differences (€)
- They have one more bit, 256 possibilities in total, that is 128 more than ASCII
- They use the same codes as ASCII

Example ISO-8859-15 and cp1252

- ISO-8859-15

character/glyph	binary(8-bit)	decimal	hex
Z	0101 1010	90	5A
{	0111 1011	123	7B
ü	11111100	252	FC
€	10100100	164	A4

- cp1252

character/glyph	binary(8-bit)	decimal	hex
Same ...			
€	10000000	128	80

Lessons Learned

- When we look at raw data (bits and bytes) we need to know(tell) the encoding to use to transform between bits and bytes and the character it is supposed to be
- Very simple for basic characters as all encodings do it the same way, here we can survive just by the rule „use default“
- 'Z' will always be '0101 1010'
- There is absolutly no bulletproof way to determine the encoding from raw data (bits and bytes)! We always have to announce the encoding we want to use!

UTF-8

- UTF-8
  - Since we need a lot more than 256 characters for all languages in the world someone had to come up with a good concept
  - UTF-8 uses up to 4-Bytes for one character
  - UTF-8 is backwards compatible with ISO-8859-15 as it only uses 1-Byte for the „basic stuff“
  - A special bit will tell wether the character is multibyte or not
  - A leading byte-sequence of ?? could/should indicate a UTF-8 encoded document
  - Encoding for unicode

Example UTF-8

- UTF-8

character/glyph	binary(unicode)	Unicode	UTF-8(hex)
Z	0101 1010	90	5A
ü		U+00FC	C3BC
€		U+20AC	E282AC

- ISO-8859-15

character/glyph	binary(8-bit)	decimal	hex
Z	0101 1010	90	5A
ü	11111100	252	FC
€	10100100	164	A4

MySQL Intro

- Good support for character sets / encoding since version 5.x
- Takes care of conversions if necessary
- default character set and available character sets determined at compile time
- Default usually is latin1*
- Utf-8 is 'utf8' ;)

* actually its cp1252!

MySQL storage related settings

- Storage
  - Encoding / Collation is set on each column, depending on type
  - Collation can only be set on columns with type varchar, text, tinytext, ???
    Does that make sense?
  - Collation set on level of database or table are only used as DEFAULTS when creating new columns, tables and no collation is supplied

MySQL connection related variables

- Variables use '_'
- Most important variables

    character_set_client

    character_set_connection

    character_set_results
- Attention!
    - Session related!
    - Difference on mysql-cli and php_mysql quite likely!
- Open questions
    - What is character_set_system used for?

```
mysql> show variables like 'character_set%';
```

MySQL charset options

- Options can be set at on cmd-line or in option-file (use '-')
- Announced on connection, used for new data, ...

  default-character-set (DEPRECATED)

  character-set-server
- Can be used to force init

  init-connect=SET NAMES `utf8`
- Skip client handshake

  Skip-character-set-client-handshake

MySQL charset „shortcuts“

- SET NAMES 'charset_name'

  SET character_set_client = x;

  SET character_set_results = x;

  SET character_set_connection = x;

- SET CHARACTER SET 'charset_name'

  SET character_set_client = x;

  SET character_set_results = x;

  SET collation_connection = @@collation_database;

  (SET character_set_connection = @@character_set_database)

Examples and Experiments

- Examples
  - I'll try and give some reallive examples
  - Change terminal encoding
  - Use mysql cli
  - Use phpmyadmin
  - Create database plus tables
  - Create,update,select data

Why the fuzz and where's TYPO3?

- MySQL modul of PHP
  - „Always“ uses latin1 as connection setting
- TYPO3
  - forceCharset = 'utf-8' makes all data within TYPO3 utf-8
  - MySQL is higly likely to <u>think</u> TYPO3 operates in latin1, unless setDBinit or some rare mysql options are used!
  - Since conversion take splace to <u>and</u> from MySQL to TYPO3 everything <u>seems</u> perfect.

Conclusion for TYPO3

- Never use 'forceCharset' without 'setDBinit' unless you reall know what you are doing!
- How can we improve/fix that for the future?

Cheers

- Rock the FLOW
- Thx and Cheers

Links

- http://en.wikipedia.org/wiki/ASCII
- http://en.wikipedia.org/wiki/ISO/IEC_8859-1#ISO-8859-1
- http://en.wikipedia.org/wiki/ISO/IEC_8859-15
- http://de.wikipedia.org/wiki/UTF-8
- http://www.utf8-zeichentabelle.de/
- http://en.wikipedia.org/wiki/Z
- http://dev.mysql.com/doc/refman/5.0/en/mysqld-option-tables.html