Collapse All | Expand All

- Home
  - Archived
  - Acknowledgements
  - Blog and News
  - Documentation v2-0-alpha
  - Documentation v1-6
  - Documentation v1-5
  - Documentation v1-1
  - Download
  - Developer
    - Coding conventions
    - Classes design v2-0
    - Database design v2-0
      - Database 2-0 Tables
    - Ticket Status Guide
    - Browser-compatibility-matrix
    - Installation notes for 1-5 development code
    - Notes on 1-0-1 fixes and stuff
    - **International Character Handling**
    - Transfering multiple files
    - HTML5
    - Testing
    - Specifications
    - release process
    - Creating repositories
    - Developers
    - Usage statistics module
    - Icons
  - Existing installations
  - Feature requests

Online Demo

Project organisation

Roadmap

Release Schedule

Release status and life cycle

Reviews and prototypes

Support and Mailinglists

Twitter

FileSender's workflows

Installation - Linux Source 2-0-Alpha-from-svn

International Character Handling

Version 5, last updated by xjansen at 2011-02-08

Assumptions

As of version beta 0.1.17 filesender can handle text in character sets other than US-ASCII. The code is based on the following assumptions:

- The Flash UI handles text as Unicode and uses UTF-8 as encoding for transmitting text strings containing international characters to the backend
- The PHP backend and database store strings with international characters in UTF-8 encoding
- Browser, Mail client and OS can handle UTF-8 encoded strings where appropriate

All handling of strings that might contain international characters should be UTF-8 aware/transparent. At the moment this concerns the 'filesubject', 'filemessage' and 'fileoriginalname' variables.

Database considerations

The *filesender* PostgreSQL database must be in either the SQL_ASCII or UTF8 encoding. Check with *psql -l*

Character set distinctions

When handling one of the relevant variables (mainly in outgoing mails and when downloading) the following distinction should be made:

1. string contains only US-ASCII characters (plain text, no special characters)
2. string contains (multibyte) UTF-8 encoded characters from the ISO-8859-1 character set only
3. string contains (multibyte) UTF-8 encoded characters from other character sets

Category 2. (UTF-8 encoded ISO-8859-1) is handled differently on some occasions to either adhere to existing standards or to prevent multibyte encoding where 'single byte encoding' can be used. In these cases the multibyte UTF-8 string is converted (iconv) to a single byte encoding in the ISO-8859-1 character set (with, where possible, the appropriate charset-labels).

Strings from category 3. (UTF-8 encoded 'other' character sets) are handled and labeled as such, UTF-8.

The "Content-Disposition" problem

At the moment the **labeling** of a string as UTF-8 encoded is not done when a file containing special characters is downloaded. The download process uses the HTTP Content-Disposition header with a filename parameter. This header is only defined for filenames containing characters from the (single byte) ISO-8859-1 character set.

Although there are currently ongoing standardisation efforts to make it possible to use other character sets there is no universally adopted way of transmitting a UTF-8 encoded filename yet (see http://greenbytes.de/tech/webdav/draftietf-httpbis-content-disp-latest.html for details). The current filesender code therefore relies on what is called 'encoding sniffing' in browsers such as FireFox, Chrome and Safari. Internet Explorer can only handle ISO-8859-1 (single byte) encoded filenames.

For now this means that when using international characters in filenames **other than ISO-8859-1** the filename will look 'funny' when downloaded with Internet Explorer and possibly other browsers other than FireFox, Chrome or Safari. The file is however correctly downloaded and usable.

Known issues

- Sometimes the Flash UI gets confused by certain international characters and will display text (also text with only ASCII characters) in a garbled way. This appears to be related to uploading filenames with characters from the Unicode 'Combining Diacritical Marks' set with Gears. This will be commonly seen with files uploaded from a Mac since the Mac HFS+ file system enforces the use of 'decomposed Unicode' characters in filenames (see http://code.google.com/p/macfuse/issues/detail?id=139#c2 for a nice explanation) . This is not affecting the functionality (uploads and dowloads use the correct text and filenames).

- Some of the PHP-code in the back-end might appear to be unneeded (most notably the utf_encode before json_decode parts).