Core - Feature # 57695

Status:	New	**Priority:**	Must have
Author:	Stephan Jorek	**Category:**	File Abstraction Layer (FAL)
Created:	2014-04-06	**Assigned To:**	
Updated:	2014-06-04	**Due date:**	
PHP Version:			
Complexity:	medium		
Sprint Focus:			
Subject:	Implement unicode normalization in TYPO3 Core's charset conversion routines, especially for filepaths in TYPO3 FAL's LocalDriver.		

Description

Preface:

If you like the following idea, I could contribute the necessary code (tests and solutions) as pull requests (as long as nobody else is interested). If I overlooked something in the Core-API and this is completely nonsense and a waste of time, then shame on me. :-)

I know that the TYPO3 Core's unicode (utf8) charset handling is quite stable and well crafted. And I don't suspect the charset-conversion to be broken, but I think its incomplete. My research is based upon commit commit:69f5be0119f9fb285d6165ab3f73aa21c3a0de53 from today and my explanations below refer to the documentation under http://www.php.net/manual/en/class.normalizer.php and the four related unicode normalization formats as mentioned in http://en.wikipedia.org/wiki/Unicode_equivalence . I'll refer to these normalization formats as NFC, NFD, NFKC and NFKD.

The Missing Feature:

Initially the following scenario seemed to be a hypothetic issue for me. As I started to move the same TYPO3 6.2 Installation between hosts with UTF8-capable filesystems based upon different unicode normalizations it happened, that the FAL's local filesystem driver reindexes files as completely new files, if their filepath unicode-normalization changes. To reproduce this behaviour one has to create a file or folder containing the german umlaut “ü” in a TYPO3-Installation on Linux/Windows and move the whole Installation to a Mac and vice-versa. Depending on the method used to transfer the files from one host to the other, a different unicode-normalization might occur, leading to different filepaths and file-identifier hashes. This makes sense as TYPO3 absolutely ignores unicode-normalization. It is currently not designed to be moved between such host constellations.

The Proposal:

If we focus on the filesystem, TYPO3 must implement at least one of the three unicode-normalization strategies (see below under *):

1. No decomposition/composition (enforced), see below under (1)
2. Normalization Form D (NFD), see below under (2)
3. Normalization Form C (NFC), see below under (3)

With the help of http://www.php.net/manual/en/class.normalizer.php and existing fallback implementations we should define and implement a normalization strategy, which at least ensures consistent filepaths across all supported platforms. Additionally we could provide commands to convert all file paths between the strategies on the supported operating- and file-systems. Finally unicode-normalization could get integrated into the charset encoding processes in general.

Strategy 1) is what we currently have, which leads to the duplicate indexing-behaviour I mentioned above. This issue needs a solution. My current assumption is that Strategy 2) and/or 3) are easier to implement.

My personal notes and research:

NFC has advantages as well as disadvantages to NFD. None of these two formats seems to be far superior, but that's another topic.

The following is given in random order:

- 1) No decomposition/composition
  - Found on most linux setups.
  - Means any canonical composition and compatibillity decomposition is supported, as nothing gets touched at all. (Really?)
  - Allows creation of visually identical looking filepaths, consisting of differently normalized pathname-fragments. Hence, in the

current FAL implementation these identical looking filepaths will probably have different file-identifier hashes => proving tests shall follow.
 - Mixed normalized paths really occured in the past (try searching the web for):
 - - older Linux Samba Machines serving Shares mounted by Mac OS X and Windows at the same time - for germans the "umlaute" like "ü" where the ones that drove some people insane
 - - (PHP-based) projects like ownClound had and still have to solve several related issues.
 - In my experience, linux software itself always produces NFC normalization, but sometimes one of the three other ones may occur. It always depends on the default-behavior of the involved filesystems, tools, clients and services.

- 2) Normalization Form D (NFD)
 - Characters are decomposed by canonical equivalence, and multiple combining characters are arranged in a specific order.
 - Mac OS X uses NFD per default and supports the three other ones. Implementations for their default encoding exists, usually its called "utf8-mac" encoding (which is actually wrong, as it is no separate encoding but a different representation).
 - Provides higher potential for really fast sorting implementations, but costs higher resource usage.

- 3) Normalization Form C (NFC)
 - Characters are decomposed and then recomposed by canonical equivalence.
 - Microsoft Windows is using NFC per default and claims to support the three other ones too. Didn't dig deeper into that.
 - This is what I suspect to be the most widespread unicode-normalization. I bet due to Windows popularity.
 - Its the W3C's recommendation for HTML5 output (and a requirement for a HTML5 compatible parser).
 - Saves some bytes, but provides less potential for blazingly fast sorting implementations. :-)

(*): Apologies for my simplifications - and no guarantee for technical correctness.

Can anyone confirm my observations ? Do you need failing unit tests ?
Does anyone have any suggestions, questions, critiques or objections ?

Cheerio,
Stephan Jorek

History

#1 - 2014-04-17 09:15 - Frans Saris

Hi Stephan,

Tnx for your research on this. This is indeed a problem. One I run into last week my self when a college uploaded some files through ftp.

Would be great if you could provide a patch for this to fix it and cover it with tests.

If you need some help with pushing a patch to our review system please let me know.

Gr. Frans

#2 - 2014-04-23 15:53 - Stephan Jorek

My current solution is a dirty hack without any tests. I'm a bit busy with other stuff, so I can't offer you something really useful for now; but approximately end of may I could take a look at this. I think i'll need some advices then, as I'm not yet familiar with the Gerrit-Flow. And I didn't sign the CLA yet.

Maybe someone else starts or even solves this tasks before end of may or I'll do so then …

Cheers,
Stephan

#3 - 2014-04-25 09:23 - Frans Saris

Hi Stephan,

could you share your "dirty hack" with us? It could be a good starting point for someone to create a patch.

gr. Frans

#4 - 2014-04-27 14:03 - Stephan Jorek

Well, I prepare my “dirty hack” to make it shareable. At least it try. Due to implementation specific issues, I'm not going to share it as it is, because I'm sure this hack raises more questions than it would solve. I'll describe some details as soon as I have something online, so please read my next posting if you are curious about this decision. And if someone really insists, because he/she needs to solve some (related) troubles right now or the like, you should contact me directly, and ask for the hack.

You can watch the implementation progress in our TYPO3.CMS fork's branch “feature-57695-fal-unicode-normalization” at github (use this link if you prefer an up-to-date diff/comparison). I strongly appreciate any feedback and will accept any helping pull-requests.

Btw.: Not a single line of code from the implementation above has ever been really tested or even run (as my original dirty hack is implemented in python and involves manipulating the “sys_file”-table from outside TYPO3). I'll mark the first working upcoming commit with “[ready for testing]”, as soon as it is “ready for testing” … :-)

Cheers,
Stephan

P.S.: I still expect (=hope) to be ready at the end of may 2014.

#5 - 2014-04-27 18:46 - Stephan Jorek

So, for now I'm stuck at the decision which strategy to implement first. Here are my current candidates:

1. Implement a converter for sys_file-entries and local filepaths respectively that forces all files to be in a certain unicode-normalization form
   - Pro: Avoids duplicate files or file-reindexing completely.
   - Con: Onetime overhead after moving TYPO3 installations between differently normalized filesystems.
   - Con: File-identifier based relations might break and must be reconstructed by some clever algorithm.
   - Con: Mixed environments (like caching-proxys or external CDNs in front of Mac OS X-based TYPO3-Servers) might cause new troubles ?
   - Hint: This is what my dirty hack currently partially solves - with python's excellent unicode support.
   - Hint: This strategy may result in a scheduler-task or an extbase-command.
   - My conclusion: I don't like this strategy, as I'm (we are ?) not able to predict all consequences, especially for the changing file-identifier.
2. Leave all filepaths as they are and force a particular unicode-normalization form just for the calculation or validation of the file-identifier.
   - Pro: Clever, smart and sounds simple. We just need to identify the best place(s) for this strategy to implement.
   - Pro: Should allow the mixed environments I mentioned above.
   - Con: If we fail to implement this strategy right, we (still) have the chance of duplicate files or file-reindexing, due to identical files which are just distinguishable by their file path's unicode-normalization. Especially Linux-based environments which usually don't care about unicode-normalization in file paths, might 'cause troubles.
   - My conclusion: This is what I'd like to go for - but not without any constructive feedback. I expect this path to consume some more time to implement, compared to the other strategy. At least writing proper initial unit tests cause headaches. :-)

Common thoughts of inspiration and open questions:

- Python and ruby both have excellent unicode-support for a while now (at least longer than any PHP-based solution I know - sorry for this dumb assumption). Do they solve this issue and if yes, then how ?
- All Java-based frameworks and projects (Alfresco, Solr, Eclipse, Spring, Play) I evaluated suffer from this bug or don't tackle this problem - although java has the same unicode-normalization support as PHP has the programmers seem not to be aware of unicode-normalization. (Write once and run everywhere has always been a lie, though … ;-)

- I remember that samba once suffered from a (possibly similar) unicode-normalization bug, which turned out to be a missing unicode-normalization feature. Maybe we can learn from their solution ? (more research needed)
- We should *not* implement a solution that assumes that the whole filesystem has unicode-normalization-awareness. We should deal with the unicode-normalization like the FAL does it with case-sensitivity for filepaths. FAL checks each file-storage for being case-sensitive or not.
  - If we would check each filepath or at least its underlying folder for being unicode-normalization-aware, we could finally compose a (local) filesystem from (real) file mounts (not those from FAL) with different normalizations. This differs from the case-sensitivity-check, I mentioned above …
  - If we would check each *file-storage* for being unicode-normalization-aware, we could finally compose a filesystem from FAL's file mounts with different normalizations. This is quite similar to the case-sensitivity-check, I mentioned above …
  - The current setting $GLOBALS['TYPO3_CONF_VARS']['SYS']['UTF8filesystem'] is not enough to reflect/configure this kind of algorithm.

Does anybody have something to add to my thoughts ? Any strategy I overlooked ? Any obvious mistake I made ?

#6 - 2014-04-27 20:41 - Stephan Jorek

Hint: my two previous comments have been updated several times. So just for the case you're reading the emails only, it is advisable to read the online versions as well.

#7 - 2014-05-10 23:34 - Stephan Jorek
- % Done changed from 0 to 70

I'd did some major progress in implementing this proposal … I added configurable unicode-normalization to FAL's LocalDriver , the TYPO3 Core's BasicFileUtility and the TYPO3 Frontend's TypoScriptFrontendController . Additionally I marked several places which still might lack unicode-normalization-awareness with proper TODO-tags or @todo-annotations.

Don't get irritated about the amount of changed files - most of them reside in typo3/contrib/Patchwork-UTF8 and belong to a pure PHP fallback-implemention I included with the help of a composer-reciept.

The current implementation still lacks the tests, proving that FAL's file-identifier-hashing fails for some TYPO3-setup-transitions when unicode-normalization is missing and does not fail anymore with the new unicode-normalization activated. The reason why I didn't implement it yet is simple: I didn't find any existing file-identifier-hashing related tests I could extend or enhance, and I was too lazy to invent a new one. (Hint: I only found file-content-hashing related tests, but that's another topic, I guess.) But I promise to implement the missing tests, as soon as I get some constructive feedback. That does not mean that there are no tests at all - there are, but none of them directly related to FAL.

So, please study the code-comments, -annotations and especially the descriptions of the additions to TYPO3's default configuration . I think/hope they contain most of my intentions. You can read them by checking out the feature-57695… branch from our repository , or what I think is easier, use the link I provided at the very beginning of this comment.

Anybody who is brave enough to crash it's TYPO3 installation is invited to give us feedback, critiques, objections or the like. Especially feedback out of the Windows-World is welcome, as I struggle with it a lot. Debian GNU/Linux, FreeBSD and OS X should to be covered quite well. But: **I still didn't test or even run anything yet**. Everything is based upon my experience or on what I have read related to this topic so far. But I'm confident that this comes close to the point where I'm officially announcing that “it's ready for testing” and of course ready for merging.

In the hope to get any feedback,
Sincerely yours,
Stephan

#8 - 2014-05-15 20:50 - Oliver Hader

Stephan, thanks for your work on this - I'm keen on having a look into your Github changes. I guess others will do this as well. So let's collect the feedback here in this issue before creating a change set for the upstream repository.

#9 - 2014-06-01 23:48 - Stephan Jorek

- % Done changed from 70 to 80

Hi Oliver,

I acknowledge your feedback-collection wish. So, I'll lead any related discussions raising at github back to this ticket here - but currently there are none. :-(

Since my last update there have been a lot of changes committed in the related feature-branch , hence I raised done state from 70% to 80%. This leaves 5% for a last personal review and 15% for any upcoming merging, pulling, pushing and reviewing. I won't add any new features to the written code from now on.

So I hope we're getting really close to point where I can shout “ready for testing”, but we're not yet there. I still didn't run any line of the code being written so far. Nevertheless, critiques, especially conception-related, are still very welcome.

Cheerio,
Stephan

P.S.: Hint - some of the previously posted hyperlinks to github-commits are outdated, so please try to translate them to the current development state by yourself.

#10 - 2014-06-04 00:38 - Stephan Jorek

Just for the record - here a (generated) list of open TODOs I'd still like to check. It might help someone to given suggestions or the like:

1. Figure out if we need/want unicode-normalization as well … in /TYPO3.CMS/typo3/sysext/backend/Classes/Controller/BackendController.php#L233
2. Figure out if we need/want unicode-normalization as well … in /TYPO3.CMS/typo3/sysext/backend/Classes/View/ThumbnailView.php#L373
3. Figure out if we need/want unicode-normalization as well … in /TYPO3.CMS/typo3/sysext/core/Classes/Charset/CharsetConverter.php#L698
4. Figure out if we need/want unicode-normalization as well … in /TYPO3.CMS/typo3/sysext/core/Classes/Charset/CharsetConverter.php#L748
5. Keep UnicodeNormalizer::filter method in sync with \Patchwork\Utf8\Bootup::filterString() in /TYPO3.CMS/typo3/sysext/core/Classes/Charset/UnicodeNormalizer.php#L219
6. Really use workaround for https://bugs.php.net/65732, means to enforce unix-linebreaks, everytime and everywhere ! in /TYPO3.CMS/typo3/sysext/core/Classes/Charset/UnicodeNormalizer.php#L222
7. Patchwork-UTF8 implementation handles cp1252 as a fallback too, but we don't do so. Is it ok for to fallback to plain utf8_encode ?!? in /TYPO3.CMS/typo3/sysext/core/Classes/Charset/UnicodeNormalizer.php#L236
8. Use UnicodeNormalizer::filterInputArrays method during core-bootstrap ? If yes, then avoid double-encoding by TSFE->convPOSTCharset ! in /TYPO3.CMS/typo3/sysext/core/Classes/Charset/UnicodeNormalizer.php#L275
9. keep method in sync with \Patchwork\Utf8\Bootup::filterRequestUri() in /TYPO3.CMS/typo3/sysext/core/Classes/Charset/UnicodeNormalizer.php#L466
10. Figure out if we need/want unicode-normalization as well … in /TYPO3.CMS/typo3/sysext/core/Classes/Http/AjaxRequestHandler.php#L77
11. Figure out if we need/want unicode-normalization as well … in /TYPO3.CMS/typo3/sysext/core/Classes/Imaging/GraphicalFunctions.php#L2662
12. ~~Figure out if we need/want unicode-normalization as well …~~ DONE Not really needed ! in /TYPO3.CMS/typo3/sysext/core/Classes/Localization/Parser/LocallangArrayParser.php#L148
13. ~~Figure out if we need/want unicode-normalization as well …~~ DONE Not really needed ! in /TYPO3.CMS/typo3/sysext/core/Classes/Localization/Parser/LocallangArrayParser.php#L155
14. Figure out if we need/want unicode-normalization as well … in /TYPO3.CMS/typo3/sysext/core/Classes/Resource/Processing/LocalImageProcessor.php#127

15. Test all unicode-normalization forms, this seems to cover NFC only, see http://forge.typo3.org/issues/57695 in /TYPO3.CMS/typo3/sysext/core/Tests/Unit/Resource/Driver/LocalDriverTest.php#L1284
16. check if “list(…) = $parts;” is really not need below in /TYPO3.CMS/typo3/sysext/frontend/Classes/Controller/TypoScriptFrontendController.php#L4709
17. Using SYS[unicodeNormalization] as fallback in FE would make sence, or not ? in /TYPO3.CMS/typo3/sysext/frontend/Classes/Controller/TypoScriptFrontendController.php#L4862
18. Different normalizations for SYS (inkl. CLI+BE) and FE could introduce new troubles ?!? in /TYPO3.CMS/typo3/sysext/frontend/Classes/Controller/TypoScriptFrontendController.php#L4863
19. Using SYS[unicodeNormalizeInputs] as fallback in FE would make sence, or not ? in /TYPO3.CMS/typo3/sysext/frontend/Classes/Controller/TypoScriptFrontendController.php#L4875
20. Different lists of input-array normalization for SYS (inkl. CLI+BE) and FE could introduce new troubles ?!? in /TYPO3.CMS/typo3/sysext/frontend/Classes/Controller/TypoScriptFrontendController.php#L4876
21. Implement SYS[unicodeNormalizeInputs] in TYPO3 bootstrap in /TYPO3.CMS/typo3/sysext/frontend/Classes/Controller/TypoScriptFrontendController.php#L4877
22. Is Unicode::normalizeInputArrays too weak … and the better approach below ? in /TYPO3.CMS/typo3/sysext/frontend/Classes/Controller/TypoScriptFrontendController.php#L4945
23. Why do we only convert $_POST charset ?!? Why not $_FILES, $_ENV, $_GET, $_POST, $_COOKIE, $_SERVER & $_REQUEST ? in /TYPO3.CMS/typo3/sysext/frontend/Classes/Controller/TypoScriptFrontendController.php#L4956
24. ~~Figure out if we need/want unicode-normalization as well …~~ DONE implemented in $basicFileFunctions above in /TYPO3.CMS/typo3/sysext/frontend/Classes/Imaging/GifBuilder.php#L705
25. Figure out if we need/want unicode-normalization as well … in /TYPO3.CMS/typo3/sysext/scheduler/Classes/Scheduler.php#L432

Good Night,
Stephan