Application: gvSIG desktop - gvSIG bugs #3959
DBF files using UTF-8 encoding are incorrectly written

12/25/2015 11:44 PM - Cesar Martinez Izquierdo

Status:	Closed	**% Done:**	0%
Priority:	Normal	**Spent time:**	0.00 hour
Assignee:			
Category:	Document view		
Target version:			
Severity:	Minor	**Add-on version:**	
gvSIG version:	2.3.0	**Add-on build:**	
gvSIG build:	2413	**Add-on resolve version:**	
Operative System:		**Add-on resolve build:**	
Keywords:		**Proyecto:**	
Has patch:	Yes	**Hito:**	
Add-on name:	Unknown		

Description

Steps to reproduce:
- Load a SHP that has a DBF encoded using UTF-8. Specify UTF-8 encoding when loading the layer
- Open the attribute table. Special characters are correctly displayed
- Start an editing session
- Modify any register
- Finish the editing session, saving the changes

Result:
The DBF is not encoded using UTF-8, thus the special characters are incorrectly encoded. The attribute table does not correctly show the special characters anymore

It happens on gvSIG 2.2.0 final and also on 2.3 devel builds.

I attach a sample UTF-8 layer for testing.

Associated revisions

Revision 43245 - 05/16/2017 08:35 AM - Joaquín del Cerro Murciano

refs #3959, Corrige y mejora sustancialmente el tratamiento del encoding de SHP/DBFs. Gracias a Cesar Martinez por el parche.

History

#1 - 10/04/2016 06:06 PM - Cesar Martinez Izquierdo

- Has patch set to Yes

- File dbf-encoding.diff added

I attach a patch that solves this problem (allowing to write SHPs/DBFs using different encodings) and also implements proper auto-detection of shp/dbf encoding when reading.

Note that it contains quite a lot of changes, so it should be carefully tested. I recommend to integrate it for the first builds of the 2.4 version, so that we have time to review the new behaviour.

Here there is a summary of all the behaviour changes:

PREVIOUS BEHAVIOUR

SHP / DBF loading

- If the user did not specify the encoding, ISO8859-1 (latin1) was used to read the layer
- If the user specified an encoding, that encoding was used to read the layer

SHP / DBF editing

- When the edition was finished & saved, the layer was always encoded using ISO8859-1 (latin1), even if a different encoding was used to load the layer (this behaviour causes the original bug described in this ticket)

SHP / DBF exporting

- There was no option to select the encoding of the new SHP (export layer)
- It was possible to select the encoding of the new DBF (export table), but it was ignored
- SHPs/DBFs were always exported using ISO8859-1 (latin1)
- The encoding field on the header was left empty

New layer (SHP) creation:

- There was no option to select encoding of SHPs
- SHPs were always created using ISO8859-1 (latin1)
- The encoding field on the header was left empty

NEW BEHAVIOUR

SHP / DBF loading

- If the user specifies an encoding, that encoding is used to read the layer
- If the user does not specify an encoding, the encoding is autodetected from dbf header if available
- If not, the encoding is autodetected from an external .cpg file if available
- As a last resort, ISO8859-1 (latin1) is used to read the layer (compatible with previous gvSIG 2.x SHPs)

SHP / DBF editing

- When the edition is finished & saved, the layer is saved using the same encoding that was used to load the layer
- The used encoding is defined on the dbf header and also on an external .cpg file (used by some dbf drivers)

SHP / DBF exporting

- There is an option to select the encoding of the new SHP. If the user selects an encoding, it is used to create the layer
- If "default" is selected, the layer will be created using UTF-8. This is generally safer than ISO8859-1 as it can encode characters from any language, **BUT** it produces some truncation effects (see below)
- The used encoding is defined on the dbf header and also on an external .cpg file (used by some dbf drivers)

New layer (SHP) creation:

- There is no option to select encoding
- SHPs are always created using UTF-8
- The used encoding is defined on the dbf header and also on an external .cpg file (used by some dbf drivers)

UTF-8 truncation effects

UTF-8 uses 1, 2, 3 or 4 bytes to store each character (variable-length encoding). This means that a DBF field of type string and maximum length of 5 will be able to store a **maximum** of 5 characters (5 dbf-ASCII characters = 5 bytes). However, less than 5 characters will be stored depending on the string to be encoded.

For instance, the string "Cesar" corresponds to 5 bytes when encoded in UTF-8, but the string "César" corresponds to 6 bytes, and "áéíóú" needs 10 bytes, and the string "ロ一マ の" (5 characters) needs 15 bytes.

In this situations, the DBF driver will truncate the string to match the maximum length defined in the field, so if length is 5:
"Cesar" will be encoded as "Cesar" (5 characters)
"César" will be truncated as "Césa" (4 characters)
"áéíóú" will be truncated as ""áé" (2 characters)
"ロ一マ の" will be truncated as "ロ" (1 character)

This might be difficult to understand by the user, which has defined a length of 5 characters.

Other GIS programs automatically modify the field length definition when writing in UTF-8 to avoid string truncation, but this might also be difficult to understand by the user (as the field definition is silently modified and might break some data model spec).

Another option would be to keep the previous gvSIG default (ISO8859-1) when exporting / creating layers, assuming that some characters (or all of them) might be lost if they are not supported by latin1.

It would also be helpful if the New Layer dialog provided a way to select the output encoding.

#2 - 03/04/2020 12:40 PM - Álvaro Anguix

- Status changed from New to Closed

Lo he testeado en el 3010 y todo funciona correctamente siguiendo los pasos indicados. Parece resuelto.

Files

turkey_cities.zip	1.33 MB	12/25/2015	Cesar Martinez Izquierdo
dbf-encoding.diff	51 KB	10/04/2016	Cesar Martinez Izquierdo