

Parsing_UTF-8_encoded_settings_file

ID	...	Creation date	July 2, 2009
Platform	S60 3rd Edition, FP1, FP2 S60 5th Edition	Tested on devices	Nokia N97
Category	Symbian	Subcategory	

Keywords (APIs, classes, methods, functions): UTF-8,ConvertToUnicodeFromUtf8

This code snippet shows how to read simple plain text settings file stored in the UTF-8 format. Settings are stored as key value pairs in the text file.

For example:

Name: Föx M?lder

Remember to add `LIBRARY charconv.lib` to your project's `.mmp` file.

Header file

```
#ifndef CUTF8FILEPARSER_H
#define CUTF8FILEPARSER_H

#include <e32std.h>
#include <e32base.h>

class CUTF8FileParser : public CBase
{
public:
//Destructor
    ~CUTF8FileParser()
//Two-phased constructor
    static CUTF8FileParser* NewL();
    static CUTF8FileParser* NewLC();

//aSettingsFileName UTF-8 encoded file to be parsed
    int ParseSettingsFileL(const TDesC& aSettingsFileName);

private:
//Splits the line into a key and value
    void ParseLine(const TDesC& aLine, TPtrC16& aKey, TPtrC16& aValue);

//Constructor for performing 1st stage construction
    CUTF8FileParser()
    void ConstructL();

private:
    ;Rfs iFs
};
#endif // CUTF8FILEPARSER_H
```

source file

```

#include <f32file.h>
#include <utf.h>

#include "UTF8FileParser.h"

//setting keys
_LIT(KVersion, "Version:");
_LIT(KText, "Text:");

_LIT8(KBOM, "\xEF\xBB\xBF"); //notepad uses bom (byte order mark) when saving as UTF-8
_LIT8(KCRLF, "\x0D\x0A"); //windows newline
_LIT8(KLF, "\x0A"); //linux newline

CUTF8FileParser::CUTF8FileParser(){}

CUTF8FileParser::~CUTF8FileParser()
{
    iFs.Close();
}

CUTF8FileParser* CUTF8FileParser::NewLC()
{
    CUTF8FileParser* self = new (ELeave) CUTF8FileParser();
    CleanupStack::PushL(self);
    self->ConstructL();
    return self;
}

CUTF8FileParser* CUTF8FileParser::NewL()
{
    CUTF8FileParser* self = CUTF8FileParser::NewLC();
    CleanupStack::Pop(); // self;
    return self;
}

void CUTF8FileParser::ConstructL()
{
    User::LeaveIfError(iFs.Connect());
}

int CUTF8FileParser::ParseSettingsFileL(const TDesC& aSettingsFileName)
{
    TInt size=0;

    TInt posNewLineDelim = KErrNotFound;

    RDebug::Print(_L("#### Parsing file %S"), &aSettingsFileName);

    RFile file;
    TInt err;
    err = file.Open(iFs, aSettingsFileName, EFileRead);

    if(err != KErrNone)
    {
        RDebug      ::Print(_L("#### Error opening descriptor file %d"), err);
        User       ::LeaveIfError(err);
    }

    CleanupClosePushL(file);
}

```

Parsing_UTF-8_encoded_settings_file

```
file.Size(size);
HBufC8 *fileContents = HBufC8::NewLC(size);

TPtr8 fileContentsPtr = fileContents->Des();
file.Read(fileContentsPtr);

//check newline used in the file http://en.wikipedia.org/wiki/Newline
//start with windows
TPtrC8 newlineDelimiter;
newlineDelimiter.Set(KCRLF);

if(fileContentsPtr.Find(newlineDelimiter) == KErrNotFound) //unix file or no newlines
{
    RDebug::Print(_L("#### Newline delimiter set to unix"));
    newlineDelimiter.Set(KLF);
}

//skip BOM if present
if(fileContentsPtr.Find(KBOM) != KErrNotFound)
{
    fileContentsPtr= fileContentsPtr.Right(fileContentsPtr.Length()-KBOM().Length());
    RDebug::Print(_L("#### bom stripped"));
    RDebug::Print(_L("#### remaining length %d"), fileContentsPtr.Length());
}

//Parse file line by line
while((posNewLineDelim = fileContentsPtr.Find(newlineDelimiter)) != KErrNotFound )
{
    RDebug::Print(_L("#### position newline %d"), posNewLineDelim);
    TPtrC8 line(fileContentsPtr.Mid(0,posNewLineDelim));

    HBufC *unicode = HBufC::NewLC(line.Length());
    TPtr16 p16 = unicode->Des();

    //convert to unicode
    CnvUtfConverter::ConvertToUnicodeFromUtf8( p16 , line );

    RDebug::Print(_L("#### Line %S"), &(*unicode));

    TPtrC16 key, value;

    //try to find key and value from the line of text
    ParseLine(*unicode, key, value);

    if(key.Length() > 0)
    {
        if(key.Compare(KVersion)==0)
        ; // do something with the version information

        if(key.Compare(KText)==0)
        ; // do something with the text
    }

    fileContentsPtr =fileContentsPtr.Right( fileContentsPtr.Length() - (posNewLineDelim+newlineD
    RDebug::Print(_L("#### remaining length %d"), fileContentsPtr.Length());
    CleanupStack::PopAndDestroy(unicode);
}

CleanupStack::PopAndDestroy(2,&file); //filecontents, file
```

Parsing_UTF-8_encoded_settings_file

```
return KErrNone;
}

void CUTF8FileParser::ParseLine(const TDesC& aLine, TPtrC16& aKey, TPtrC16& aValue)
{
    //one can easily implement comment option by skipping
    //lines starting with specific character

    TLex lexer(aLine);

    lexer.SkipSpace(); //skip start space

    lexer.Mark();
    lexer.SkipCharacters();

    aKey.Set(lexer.MarkedToken()); //key

    lexer.SkipSpaceAndMark();
    aValue.Set(lexer.Remainder()); //value
}
```