Subject: Re: Best Unicode strategy
Posted by mirek on Sat, 12 Nov 2011 11:09:05 GMT
View Forum Message <> Reply to Message

conrad wrote on Sat, 12 November 2011 03:34Case:
Database access via a managed c++ wrapper package to .Net assemblies. In the .Net universe Strings are utf16.

As I understand it, U++ is by default "unicode aware" by virtue of its widgets understanding utf8. A simple test revealed that I can feed a mixed English & Chinese char* to PromptOK like:

Since most U++ text widgets take in String (not WString) I need to
make a decision as to what to transport.

As I see it there are 2 options:
1. Use String and pay the cost for many needed string manipulations since String itself is not really utf8 aware. I.o.w. many temp conversions to WString and back.
2. Use WString and convert to a String (utf8 somehow - not with ToString() since that uses a character set) before passing on to U++ widgets/drawing routines.

I'd like to hear what strategy other user of U++ have used?
I wonder if I have missed out on something or not dug deep enough.

Conrad

My default strategy is to use utf-8 (or default charset, for legacy) String everywhere and only convert to WString if necessary; this involves either manipulation with individual characters (in that situation, I convert String->WString, do the operation, then WString->String) or dealing with 3rd party (your case).

String itself is much more efficient that WString in many situations, like storage, comparison, copy, maps. Dealing with database library, you are likely to spend much more time in database than by converting String->WString.

BTW, ToString (and ToWString) is perfectly adequate too (it converts from/to default charset, which, by default, is utf-8).

Mirek