

# CMultiLanguageDialog & rc\_translate

## Was können diese Routinen?

Diese Routinen ermöglichen es dem Programmierer einfach mehrsprachige Programme mit Visual C++ 6.0 und neuer (hoffentlich) zu erzeugen. Zuerst wird die Systemsprache ermittelt, falls die Texte in dieser vorhanden sind wird sie verwendet, sonst Englisch oder die Originalsprache des Programms. Aber der Nutzer kann die Sprache auch jederzeit ändern. Im Moment werden Dialog-Boxen, Menüs und Stringtables übersetzt. Die Routinen wachen bei Bedarf und entstehen für meine Programme, was ich da also nicht brauche ist auch noch nicht drin. Wer es trotzdem für andere Programme nutzen will kann dies machen muss aber ggf. fehlendes ergänzen. Auch ist dies kein Programm für den Endnutzer, ich gehe davon aus, dass es von Programmierern benutzt wird die im Notfall selber Einstellungen ändern können und Fehler per Debugger finden.

## Wie erzeugt man ein mehrsprachiges Programm mit diesen Routinen?

1. CMultiLanguageDialog.h und CMultiLanguageDialog.cpp dem Projekt hinzufügen.
2. Bei der Deklaration aller Dialoge CDialog durch CMultiLanguageDialog ersetzen, dabei natürlich bei Bedarf  
#include "CMultiLanguageDialog.h"  
einfügen. Ersetzt werden muss es in:
  1. Klassendeklaration, z.B.  
class CAboutDlg : public CDialog
  2. CRc\_translateDlg::CRc\_translateDlg(CWnd\* pParent /\*=NULL\*/) : CDialog(CRc\_translateDlg::IDD, pParent)  
versucht man dann zu compilieren erhält man ggf. C2614 – Fehler. Dort auch jeweils CDialog durch CMultiLanguageDialog ersetzen.
3. Alle  
Cdialog::OnInitDialog();  
durch  
CmultiLanguageDialog::OnInitDialog();  
ersetzen.
4. Verwaltung der aktuellen Sprache: zur Sprachauswahl kann man z.B. ein erweitertes Kombinationsfeld benutzen unter Data trägt man nur \_LANGUAGE\_ ein, dies wird dann durch die Sprachen im HTML-File ersetzt. Dadurch kann der User eine neue Sprache hinzufügen, ohne dass das Programm geändert werden muss. Per Klassenassistent weist man diesem Kombinationsfeld eine Member-int-Variable zu, z.B. m\_nLang. Diese wird hinter dem Aufruf von CmultiLanguageDialog::OnInitDialog() gleich zu getLang() gesetzt, z.B.:  
m\_nLang=getLang();  
zumindest wenn man die Standardinitialisierung übernehmen will und nicht z.B. die Sprache aus einem Konfigurationsfile ausliest, sonst muss man entsprechend  
adjustLanguage(m\_nLang );  
aufrufen. Per Doppelklick auf dieses Kombinationsfeld im Resourceeditor fügt man eine Behandlungsroutine für eine Selektionsänderung hinzu, z.B.:  
void CRc\_translateDlg::OnEditchangeLang()  
{  
UpdateData(TRUE);  
adjustLanguage(m\_nLang );  
}

5. nicht zu übersetzende Texte: nicht alles soll übersetzt werden, vor allem nicht Texte die das Programm selber ausgibt, z.B. solche die einer Member-Variablen zugeordnet werden. Derzeit kann man ein Übersetzen nur verhindern indem der Text nicht im html enthalten ist. Also notfalls die Zeile löschen. Allerdings wird sie dann beim nächsten Lauf von rc\_translate (s.6.) wieder angelegt. Damit dies nicht geschieht muss der Text im rc-File auf „Static“, also der Voreinstellung bleiben. Solche Texte werden von rc\_translate ignoriert
6. Erstellen des html-Files mit den verschiedenen Sprachen. Hierzu dient das beiliegende Programm rc\_translate. Um die nötigen Tabellen zu erzeugen braucht es als Quelle das rc-File des Projekts und einige Headerfiles:
  1. resource.h: sollte dort sein wo auch das rc-File ist und automatisch gefunden werden.
  2. winuser.h
  3. afxres.h: Die beiden letzten sind Standardincludefiles von Visual C++. Das Programm versucht sie automatisch zu finden, falls das nicht funktioniert muss man sie per Hand hinzufügen.

Existiert das html-File bereits wird es ergänzt, gemachte Eingaben bleiben also erhalten. Außerdem wird noch eine Liste aller Projekte erstellt. Dies dient als Wörterbuch, hat man also den gleichen Text schon mal irgendwo verwendet wird er automatisch übersetzt. Mit OK wird das html-File erstellt. Es enthält 3 Tabellen, von diesen sollte man nur die erste bearbeiten!

## Bearbeitung des html-Files

Als Beispiel benutze ich das Programm rc\_translate.

Bearbeiten kann man es z.B. mit OpenOffice.Writer. Wie gesagt sollte man nur die erste Tabelle bearbeiten, und hier nur die Spalten ab „Status“. Diese enthält die Info ob sich etwas geändert hat. Ein neuer Text (im rc enthalten, im alten html nicht) wird mit „NEW“ gekennzeichnet. Hat sich der Text geändert erhält er ein „CHG“ ein Text der im rc nicht mehr vorhanden ist ein „DEL“ Ein einmal gesetzter Status bleibt ewig erhalten, man muss ihn also selber löschen wenn man die Zeile bearbeitet hat. Durch diese Spalte kann man schnell finden wo man was eingeben muss. Die nächste Spalte enthält die Originalsprache, Änderungen hier werden beim nächsten Lauf von rc\_translate überschrieben, die Originalsprache sollte man also in Resourceeditor und nicht im html ändern. Keine Regel ohne Ausnahme. Die Sprachennamen kann man hier auf Dauer ändern s.u.

Will man eine neue Sprache hinzufügen muss man 5 Sachen machen:

1. Einen neue Spalte hinzufügen
2. im obersten Feld dieser Spalte den Namen der Sprache in einer beliebigen Sprache einfügen.
3. Im zweiten Feld die ID dieser Sprache eintragen, s. <[http://msdn.microsoft.com/en-us/library/dd318693\(v=VS.85\).aspx](http://msdn.microsoft.com/en-us/library/dd318693(v=VS.85).aspx)> Man kann entweder den Wert aus der Spalte „Prim.lang. Identifier“ oder den aus „Locale identifier“ benutzen. Letzteres ist nur sinnvoll wenn man mehrere Länderversionen einer Sprache einfügt. Wichtig ist die ID nur zur automatischen Sprachauswahl, und 2 Spalten dürfen nicht die gleiche ID haben.
4. Für die Sprachauswahl braucht man noch den Sprachnamen in allen bereits vorhandenen Sprachen, Angenommen Sie wollten als 4. Sprache Spanisch hinzufügen so müsste die folgende Zeile eingefügt werden:

IDD_RC_TRANSLATE_DIALOG(ID C_LANG ):4	L	4	0x6603e9		Spanisch	Spanish	espanol
--	---	---	----------	--	----------	---------	---------

Das erste Feld ist dabei unwichtig. Die ID im 4. Feld könnte sich in Zukunft ändern, falls es also nicht klappt mal die anderen Zeilen mit den Sprachen suchen und vergleichen.

5. In den restlichen Feldern der neuen Spalte die Texte übersetzen. Dabei ist die ID STRING\_TABLE(IDS\_HLPFILE) etwas spezielles, hier geht es nicht um die Übersetzung eines Textes, sondern um den Filenamen des Helpfiles. Hat man also neben der Übersetzung des Programms auch das Helpfile ins Spanische übersetzt und es z.B. ayuda.pdf genannt, so müsste dies hier eingetragen werden. Hat man es nicht übersetzt und will stattdessen die deutsche Hilfe angezeigt bekommen trägt man deren Namen also liesmich.pdf ein.

Beim Programmlauf wird immer versucht den Text in der aktuellen Sprache zu verwenden, ist dieser nicht gesetzt wird

### Rechtliches und Quelle:

rc\_translate steht unter der GPL V3 Lizenz.

<http://www.gnu.de/documents/gpl.de.html>

Die Files CMultiLanguageDialog.h und CmultiLanguageDialog.cpp können alternativ zu einer lib compiliert werden und dann unter der LGPL V3 Lizenz benutzt werden:

<http://www.gnu.de/documents/lgpl-3.0.de.html>

Meine aktuelle Version sollte hier zu finden sein:

[http://www.dimitri-junker.de/html/software\\_\\_pc.html](http://www.dimitri-junker.de/html/software__pc.html)

Aber da es von anderer Seite weiterentwickelt worden sein könnte muss Dies nicht die aktuelle Version sein. Deshalb bitte ich alle, die an dem Programm arbeiten, mir dies mitzuteilen, damit es nur eine Version gibt.

Dimitri Junker