Plusieurs langues pour son exe

Pour cet exemple, je vous épargne le discours sur l'internationalisation déjà vu en Java car c'est le même concept.

La première étape est de créer un fichier de ressource (nom_du_fichier.rc) où nous écrivons les chaînes de caractères de toutes les langues (voir exemple ci-dessous), chaque chaîne nous lui associons une clef de type entier. La subtilité réside dans le fait où il faudra trouver le moyen pour distinguer les langues. Pour ma part, j’utilise cette technique qui consiste à donner à chaque langue un identifiant (1 pour le français et 2 pour l’anglais) et le même identifiant pour les chaînes identiques (001 pour "Fichier" et 001 pour "File"), cette technique à une limite elle ne nous permet pas d’avoir plus de 999 chaînes de caractères par langues.Langues.rc

```
STRINGTABLE
BEGIN
1001, "Fichier"
1002, "Quitter"
1003, "Outils"
1004, "Langues"
1005, "Anglais"
1006, "Français"
1007, "A propos"
2001, "&File"
2002, "&Exit"
2003, "&Tools"
2004, "&Language"
2005, "&English"
2006, "&French"
2007, "A bout"
END
```

Il faut ensuite générer un fichier de type ".RES" (fichier de ressource pour Delphi) en utilisant l'utilitaire fourni par Borland "BRCC32" (disponible dans le répertoire "bin" de Delphi) et en tapant la commande suivante : "BRCC32 Langues.rc". Le résultat de cette commande c'est un fichier "Langues.RES" que nous devons déclarer dans notre application en ajoutant la directive {$R Langues.RES} après l'implémentation de la fenêtre.Form1: TForm1;

```
implementation
{$R Langues.RES}
{$R *.dfm}
```

Ce qui reste à faire est de récupérer les chaînes en fonction de la langue, pour cela nous utilisons la fonction LoadStr, par exemple : Fichier1.Caption:=LoadStr(id*1000+1);si la langue est le français id=1 donc la clé= 1001 sinon la clé=2001 qui corresponde respectivement à "Ficher" et "File".

Exemple à télécharger : Internationalisation.zip