# DFM-updater

Sommaire

[DFM-updater 1](#_Toc433876509)

[Objectif 1](#_Toc433876510)

[Compilation et exécution 1](#_Toc433876511)

[Fonctionnement 1](#_Toc433876512)

[Exemple : Modifier la valeur d’une propriété du DFM 2](#_Toc433876513)

[Les règles de conversion existantes 3](#_Toc433876514)

## Objectif

automatiser les mises à jour répétitives sur les Forms ou le code d’une application C++ Builder. Exemple :

* Ajouter/modifier/supprimer des propriétés d’un contrôle
* Renommer des fonctions, variables
* Modifier le nombre ou le type des paramètres d’une fonction
* Ajouter de nouveaux évènements d’évènements
* Ajouter/modifier/supprimer du code C++
* Modifier l’apparence d’un type de contrôle
* Etc.

## Compilation et exécution

Le code JAVA est dans <https://srv-dev/svn/Génésis/trunk/Outils/DFM-updater/DFM-updater>.

C‘est un projet Maven, développé avec Eclipse Mars.

L’application :

* Peut être compilée avec le script package.bat
* Peut être lancée avec le script run.bat. Ex :

|  |
| --- |
| Run.bat c:\Borland\RH\_TRUNK\\*.dfm |

## Fonctionnement

La classe RhUiModernizer est le point d’entrée.

La méthode RhUiModernizer.run() énumère les fichiers passés en paramètre et traite chacun d’eux avec la méthode ProcessDfm().

ProcessDfm() utilise la classe DfmReaderWriter pour parser le fichier DFM et créer un arbre des contrôles et de leurs propriétés, modélisé par la classe DfmObject.

* Le format des DFMs dans Builder 4 étant binaire et propriétaire, DfmReaderWriter utilise l’outil convert.exe de Borland pour convertir du binaire vers du texte, et vice versa.
* L’emplacement de convert.exe est défini dans DfmReaderWriter.convert().

Le code C++ est parsé (sommairement) par la classe CppReaderWriter.

* Le code C++ est reformaté à l’aide de l’outil Astyle.exe, invoqué par CppReaderWrite.reformat().

La métode RhUiModernizer.updateDfmObjects() se charge de modifier le contenu des DFMs. Elle s’appuie sur une liste de règles de conversion, chaque règle dérivant de la classe AConversionRule.

Pour effectuer un traitement, une règle existante est réutilisée (ex : ChangePropertyValue permet de modifier la valeur d’une propriété d’un objet du DFM), ou une nouvelle règle dérivant de AConversionRule est créée.

Les méthodes abstraites de AConversionRule à implémenter :

* isApplicable() renvoie true si la règle doit être appliquée à l’un des objets passés en paramètre (mais pas à leurs enfants, la méthode est appelée récursivement pour chaque enfant).
* doApply() applique les modifications aux objets passés en paramètre (mais pas à leurs enfants, la méthode est appelée récursivement pour chaque enfant)

La métode RhUiModernizer.updateCppCode() se charge de modifier les fichiers cpp et h. Elle s’appuie également sur une liste de règles de conversion dérivant de AConversionRule.

## Exemple : Modifier la valeur d’une propriété du DFM

On souhaite modifier la valeur de la propriété BorderColor de tous les objets TColoredGroupBox.

Une classe dérivée de AConversionRule doit implémenter la fonctionnalité : ChangePropertyValue.

ChangePropertyValue doit être générique pour être applicable à d’autres classes et d’autres propriétés. Elle doit prendre en paramètre le nom de la classe des objets dont une propriété doit être modifiée, le nom de la propriété affectée, et la nouvelle valeur à utiliser.

La méthode isApplicable() de ChangePropertyValue doit renvoyer true si la classe de l’objet passé en paramètre est celle à traiter.

La méthode doApply() se chargera de remplacer la valeur de la propriété pour l’objet passé en paramètre :

|  |
| --- |
| package conversion;  import cpp.CppClass;  import dfm.DfmObject;  public class ChangePropertyValue extends AConversionRule {  String objectTypeRegEx;  String propName;  String propValue;    public ChangePropertyValue(String anObjectTypeRegEx,  String aPropName,  String aPropValue) {  objectTypeRegEx = anObjectTypeRegEx;  propName = aPropName;  propValue = aPropValue;  }    @Override  public boolean isApplicable(DfmObject dfmObject, CppClass cppClass) {  String typeName = dfmObject.getTypeName();  return typeName.matches(objectTypeRegEx);  }  @Override  protected boolean doApply(DfmObject dfmObject, CppClass cppClass) {  dfmObject.properties().put(propName, propValue);  return true;  }  } |

Comme seul le fichier DFM est affecté (les .h et .cpp ne seront pas modifiés), seule la méthode RhUiModernizer.updateDfmObjects() doit être mise à jour avec la nouvelle règle :

|  |
| --- |
| **private** **boolean** updateDfmObjects(DfmObject dfmObject,  CppClass cppClass) {  ArrayList<AConversionRule> rules = **new** ArrayList<AConversionRule>();  rules.add(**new** ChangePropertyValue("TColoredGroupBox","BorderColor","15189669"));  … |

## Les règles de conversion existantes

|  |  |
| --- | --- |
| Classe | Utilité |
| AddInclude | Ajoute un include à un fichier s’il n’est pas déjà présent, et si la condition passée en paramètre est vérifiée. |
| ChangeBaseClass | Modifie le type de la classe de base dont dérive le DFM (pour dériver de TFormExtented au lieu de TForm) |
| ChangeFont | Remplace la police utilisée pour un type d’objet |
| ChangeObjectType | Change le type d’un objet (ex : pour convertir les TEdit en TColoredEdit, le TGroupBox en TColoredGroupBox) |
| ChangePropertyValue | Change la valeur de la propriété d’un type d’objet (ex : la propriété BorderColor des TColoredGroupBox) |
| CompositeRule | Permet de regrouper plusieurs règles en une seule (ex : pour convertir les TEdit en TColoredEdit, et pour ajouter en même temps les include nécessaire) |
| FixSpreadSetSetParams | Caste en TVariant le 3ème paramètre des appels à la méthode TfpSpread ::SetText() |
| RemoveLineOfCode | Supprime les lignes contenant le code C++ spécifié |
| RemoveProperty | Supprime une propriété d’un type d’objet dans le DFM |
| RenameProperty | Renomme une propriété d’un type d’objet dans le DFM |
| ReplaceCode | Remplace le code C++ spécifié par un autre |
| RestyleBevel | Ajuste les Anchors des TBevel pour gérer les redimensionnements |
| RestyleForm | Modifie les propriétés des TForm en fonction du nouveau design (couleur, bordures, police, etc.) |
| RestyleSpeedButton | Ajuste les propriétés des TSpeedButton (taille, police, ancrages, icône, alignement) en fonction de leur position sur la form et de leur libellé |
| RestyleSpreadPanel | Identifie les groupes de composant correspondant au modèle de Spread (TfpSpread + barre de boutons + image en haut à droite) et les relooke conformément au nouveu design (couleur, icônes, ancrages) |
| TvaToTfpSpread | Convertit les TvaSpread en TfpSpread |
| UseParentFont | Met la propriété « UsePArentFont » à true pour les composants qui utilisent la même police que le TForm parent. |