Mit 'Babysteps' in die XTEXT-XTEND Welt

Teil 3: ein externer Codegenerator für Copy&Paste

Anbei noch einmal die Code Listings wie sie der Editor zeigt, also mit Syntax Highlites. Passend zum Artikel im Eclipse Magazin.

von Ulrich Merkel

Grammatik: umecode2.xtext

```
grammar de.ulrichmerkel.Umecode2 with
org.eclipse.xtext.common.Terminals

generate umecode2 "http://www.ulrichmerkel.de/Umecode2"

Model:
      {Model}
    ('generator_options' options+=Option? (',' options+=Option)*)*
      ('componentvariables=' componentvariables+=Variable? (','
componentvariables+=Variable)*)?
      ('globalvariables='    globalvariables+=Variable? (','
globalvariables+=Variable)*)?
      entities+= Entity*
      modules+=Module*
;
Option:     name=ID '=' value=STRING;
Entity:
      'entity' name=ID '.' model=ID aDescription=ADescription?
aComment=AComment? ('painted_fields=' painted_fields+=Field? (','
painted_fields+=Field)*)?
      'fields=' fields += Field? (',' fields+= Field)*
      ('primarykeyfields=' primarykeyFields+=PKField? (','
primarykeyFields+=PKField)*)?

;
QualifiedName:
    ID ('.' ID)*;

Variable:
      name=ID
;
Field:
      name=ID
;
PKField:
      name=ID
;
AComment: {AComment}
      'comment=' commentText=STRING?
;
ATodo: {ATodo}
      'todo=' todoText=STRING?
;
ADescription: {ADescription}
      'description=' descriptionText=STRING?
;
ALpmx: {ALpmx}
      'LPMX=' text=ML_FREECODE?
      ;

Module:     (FreeCode | Trigger
      | Arefentity | Arefentityfield | ATodo | Mergefile |
SupportModule
```

```
      | DefinePair | DefinePairStringString | DefinePairString
);
DefinePair: '#definepair' name=ID defpairs+=Id2Id (','
defpairs+=Id2Id)*;
DefinePairString: '#definepair_string' name=ID defpairs+=Id2String
(',' defpairs+=Id2String)*;
DefinePairStringString: '#definepair_string_string' name=ID
defpairs+=String2String (',' defpairs+=String2String)*;
Id2Id: idkey=ID '=' idvalue=ID;
Id2String: idkey=ID '=' idvalue=STRING;
String2String: idkey=STRING '=' idvalue=STRING;

Trigger:    'trigger' name=('LPMX'|'DEFN'|'ABCD') text=ML_FREECODE;
Arefentity: 'arefentity' name=ID useent=[Entity];
Arefentityfield: 'arefentityfield' name=(ID|'<$fieldname>')
useentfield=[Field|QualifiedName];
FreeCode:   text=ML_FREECODE;
Mergefile:  'mergefile' name=ID ('values=' values+=KeyValuePair?
(',' values+=KeyValuePair)*)*;
KeyValuePair:     name=ID '=' value=STRING;
SupportModule: 'workentities=' workentities+=[Entity] (','
workentities+=[Entity])*;

   terminal ML_FREECODE :    '«' -> '»'  ;
```

Generator: umecode2Generator.xtend

```
/*
 * generated by Xtext
 */
package de.ulrichmerkel.generator

import de.ulrichmerkel.umecode2.ATodo
import de.ulrichmerkel.umecode2.Arefentity
import de.ulrichmerkel.umecode2.Arefentityfield
import de.ulrichmerkel.umecode2.DefinePair
import de.ulrichmerkel.umecode2.DefinePairString
import de.ulrichmerkel.umecode2.DefinePairStringString
import de.ulrichmerkel.umecode2.Entity
import de.ulrichmerkel.umecode2.FreeCode
import de.ulrichmerkel.umecode2.Id2Id
import de.ulrichmerkel.umecode2.Id2String
import de.ulrichmerkel.umecode2.KeyValuePair
import de.ulrichmerkel.umecode2.Mergefile
import de.ulrichmerkel.umecode2.Model
import de.ulrichmerkel.umecode2.Module
import de.ulrichmerkel.umecode2.Option
import de.ulrichmerkel.umecode2.String2String
import de.ulrichmerkel.umecode2.SupportModule
import de.ulrichmerkel.umecode2.Trigger
import java.awt.Toolkit
import java.awt.datatransfer.Clipboard
import java.awt.datatransfer.StringSelection
import java.io.File
import java.io.FileReader
import java.util.Collection
import java.util.HashMap
import java.util.LinkedList
import java.util.Map
import java.util.TreeMap
import java.util.regex.Matcher
import java.util.regex.Pattern
import org.eclipse.emf.common.util.EList
import org.eclipse.emf.ecore.EObject
```

```
import org.eclipse.emf.ecore.resource.Resource
import org.eclipse.xtext.generator.IFileSystemAccess
import org.eclipse.xtext.generator.IGenerator

import static extension com.google.common.io.CharStreams.*

/**
 * Generates code from your model files on save.
 *
 * see
http://www.eclipse.org/Xtext/documentation.html#TutorialCodeGenerat
ion
 */
class Umecode2Generator implements IGenerator {



	def setClipboard(String workstring) {
		try {
			var Toolkit toolkit =
Toolkit.getDefaultToolkit();
			var Clipboard clipboard =
toolkit.getSystemClipboard();
			var StringSelection strSel = new
StringSelection(workstring);
			clipboard.setContents(strSel, null);
		} catch (Exception e) {
			e.printStackTrace();
		}
		return workstring
	}
	// *** advanced Option utilities
	static Map<String, String> myOptions = new TreeMap<String,
String>(String.CASE_INSENSITIVE_ORDER);

	def loadMyOptions(EList<Option> options) {
		myOptions.clear
		for (Option option : options) {
			myOptions.put(option.name, option.value)
		}
		return ("")
	}

	def getMyOption(String key) {
		return myOptions.get(key) ?: ""
	}

	def boolean isMyOption(String key) {
		return myOptions.containsKey(key)
	}

	def boolean isMyOption(String key, String compareValue) {
		compareValue.equals(getMyOption(key))
	}

	def boolean isMyOptionIgnoreCase(String key, String
compareValue) {
		compareValue.equalsIgnoreCase(getMyOption(key))
	}

	// *** Collections to handle log and error messages
	Collection<String> logLines = new LinkedList();

	def addLog(String str) {
```

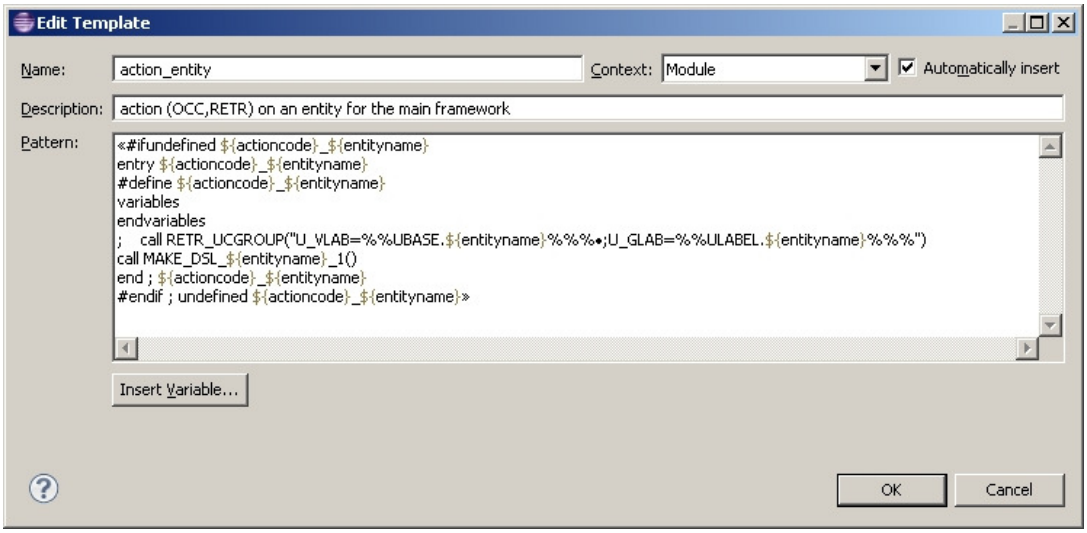
```
			logLines.add(str);
			return ""
		}

		def reportLog() '''«IF (logLines.size() > 0)»
; *** Log-Information:
«FOR String s : logLines»
«s»
«ENDFOR»
«ENDIF»'''

		def resetLog() {
			logLines.clear;
			return ""
		}

		def addIncludeFile(String filename) '''
«val File incfile = new File("..\\!includefiles\\" + filename)»
«IF incfile.exists()»
«FOR String line : new FileReader(incfile).readLines»
		«line»
«ENDFOR»
«ELSE»
«addLog("Error: addIncludeFile: >" + filename + "< not in
!includefiles directory")»
«ENDIF»
'''

		def replaceMergeVariables(EList<KeyValuePair> preferences,
String mergeText, String fileID) {
			var mergeTextWork = mergeText;
			if (mergeText != null) {
				for (KeyValuePair kvp : preferences) {
					mergeTextWork =
mergeTextWork.replaceAll("\\$\\{" + kvp.name + "\\}", kvp.value);
				}
				checkForPlaceholders(mergeTextWork, fileID)
				return mergeTextWork
			} else
				return ""
		}

		def checkForPlaceholders(String workText, String fileID) {
			checkForPlaceholderNames(workText, fileID)
			checkForPlaceholderNames1(workText, fileID)

			//	if (workText.contains("${")) {addLog("Error:
Still unresolved placeholders in >" + fileID + "<: \n>"
+workText+"<")}
			return ""
		}

		def checkForPlaceholderNames(String workText, String fileID)
{
			var String[] places = workText.split("\\$\\{")
			var HashMap<String, String> hashNames = new HashMap()
			var int pos = places.length - 1;
			if (workText.contains("${")) {
				while (pos > 0) {
					var String place = places.get(pos)
					hashNames.put(place.substring(0,
place.indexOf("}")), "")
					pos = pos - 1
				}
```

```
                  addLog(
                        "Warning: " + hashNames.size + " unresolved
placeholders in " + fileID + ": " +
                              hashNames.keySet.join(", "))
            }
            return ""
      }

      def checkForPlaceholderNames1(String workText, String fileID)
{
            var HashMap<String, String> hashNames = new HashMap()
            if (workText.contains("${")) {
                  var Pattern pattern =
Pattern.compile("\\$\\{([^}]*)\\}");
                  var Matcher matcher = pattern.matcher(workText);
                  while (matcher.find()) {
                        hashNames.put(matcher.group(1), "")
                  }
                  addLog(
                        "Warning1: " + hashNames.size + "
unresolved placeholders in " + fileID + ": " +
                              hashNames.keySet.join(", ")
                  )
            }
            return ""
      }
      /* now the main thing */
      override doGenerate(Resource resource, IFileSystemAccess fsa)
{
            var String scriptFileName =
resource.getURI().lastSegment.toString();
            var String outFileBase = scriptFileName.substring(0,
scriptFileName.lastIndexOf("."));
            fsa.generateFile(outFileBase + '.2.CODE',
makeTheCodeMain(resource.contents.head as Model))

            // *** output of log and error messages
            if (logLines.size() > 0)
                  fsa.generateFile(outFileBase + '.LOG',
reportLog());
      }

def makeTheCodeMain(Model sm) '''
«setClipboard(makeTheCode(sm).toString)»
'''
      def makeTheCode(Model sm) '''
«FOR Module modul : sm.modules»
«switch modul {
            FreeCode: {
                  genFreeCode(modul)
            }
            Trigger: {
                  genTriggerCode(modul)
            }
            Arefentity: {
                  genArefentityCode(modul)
            }
            Arefentityfield: {
                  genArefentityfieldCode(modul)
            }
            ATodo: {
                  genAtodo(modul)
            }
            Mergefile: {
```

```
                addMergeFile(modul)
            }
            SupportModule: {
                genSupportModules(modul)
            }
            DefinePair: { genDefinePair(modul) }
            DefinePairString: { genDefinePairString(modul) }
            DefinePairStringString: {
genDefinePairStringString(modul) }
      }»
«ENDFOR»
      '''

def util_genDefinePair(String pairName, String pairKeys, String
pairValues) {'''
;generated pairs
#define «pairName»_k = «pairKeys»
#define «pairName»_v = «pairValues»
#if copy = use_pairdefines_template
#for «pairName»_k1 = (<«pairName»_k>),«pairName»_i
; loop to find the matching partner
#for «pairName»_v1 = (<«pairName»_v>),«pairName»_i1
#if <«pairName»_i> = <«pairName»_i1>
;message $concat("<«pairName»_i>:","<«pairName»_k1> points to
<«pairName»_v1>")
; here you can work with both parts of the pair
#endif
#endfor ; «pairName»_v1
#undefine «pairName»_i1
#undefine «pairName»_v1
#endfor ; «pairName»_k1
#undefine «pairName»_i
#undefine «pairName»_k1
#endif copy = use_pairdefines_template
'''
}
def genDefinePair(DefinePair pair) {
      util_genDefinePair(pair.name

      ,pair.defpairs.filter(typeof(Id2Id)).map[idkey].join(',')

      ,pair.defpairs.filter(typeof(Id2Id)).map[idvalue].join(',')
      )
}

def genDefinePairString(DefinePairString pair) {
      util_genDefinePair(pair.name

      ,pair.defpairs.filter(typeof(Id2String)).map[idkey].join(',')

      ,pair.defpairs.filter(typeof(Id2String)).map[idvalue].join(',
')
      )
}

def genDefinePairStringString(DefinePairStringString pair) {
      util_genDefinePair(pair.name

      ,pair.defpairs.filter(typeof(String2String)).map[idkey].join(
',')

      ,pair.defpairs.filter(typeof(String2String)).map[idvalue].joi
n(',')
      )
```

```
}


      def genSupportModules(SupportModule smodule) '''

;*************** support routines **************
#for workentity = («smodule.workentities.map[name].join(', ')»)
«addIncludeFile("umecode1_Supportmodules.txt")»
#endfor ; workentity
#undefine workentity
'''

      def addMergeFile(Mergefile mf)
'''«replaceMergeVariables(mf.values, addIncludeFile(mf.name +
".txt").toString,
            mf.name + ".txt")»'''

      def genAtodo(ATodo modul) '''
;
;@TODO
;«escapeStringToUComment(modul.todoText)»
;
'''

      def genArefentityCode(Arefentity modul) '''
Refentity «modul.name» «modul.useent.name».«modul.useent.model» ;
full qualified Entity
'''

      def genArefentityfieldCode(Arefentityfield modul) '''
Refentityfield «modul.name»
«modul.useentfield.name».«genParentname(modul.useentfield.eContaine
r())» ; full qualified Field
'''

      def genParentname(EObject container) '''«switch container {
            Entity: {return container.name}
      }»
'''

      def genTriggerCode(Trigger modul) '''
Trigger Code for «modul.name»
«decodeFreeCode(modul.text)»
'''

      def genFreeCode(FreeCode modul) '''
«decodeFreeCode(modul.text)»
'''

      def String decodeFreeCode(String freeCode) {
            freeCode.substring(1, freeCode.length - 1)
      }


      def escapeStringToUComment(String string) {
            if(string != null) string.replaceAll('\r\n', ';\r\n')
      }

   }
```

BILDÜBERSICHT



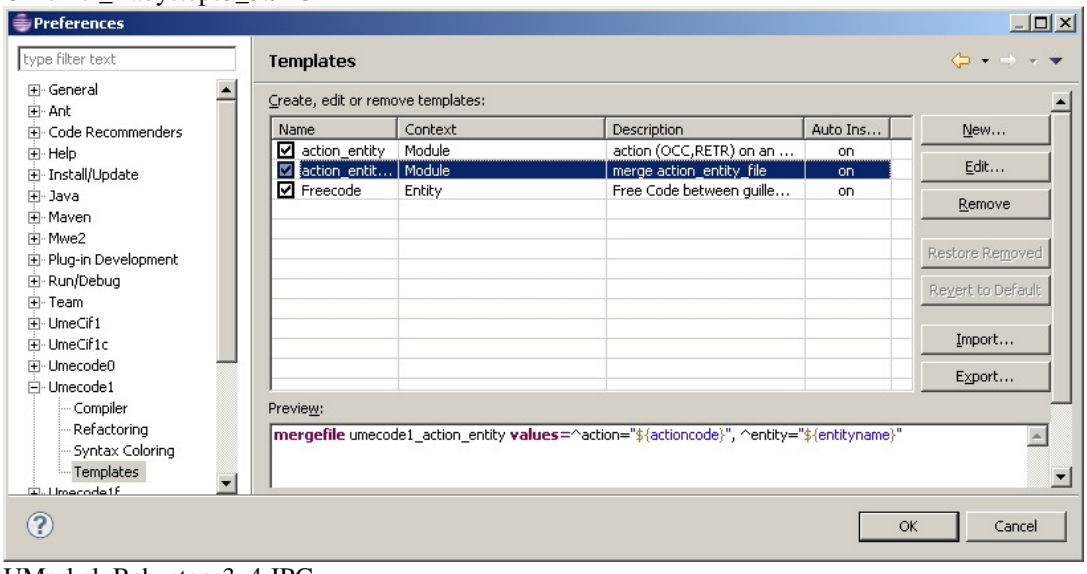
UMerkel_Babysteps3_1.JPG

```
«#ifundefined OCC_entityname
entry OCC_entityname
#define OCC_entityname
variables
endvariables
;    call RETR_UCGROUP("U_VLAB=%%UBASE.entityname%%%•;U_GLAB=%%ULABEL.entityname%%%")
call MAKE_DSL_entityname_1()
end ; OCC_entityname
#endif ; undefined OCC_entityname»
```

UMerkel_Babysteps3_2.JPG

```
«#ifundefined OCC_ENTITY2
entry OCC_ENTITY2
#define OCC_ENTITY2
variables
endvariables
;    call RETR_UCGROUP("U_VLAB=%%UBASE.ENTITY2%%%•;U_GLAB=%%ULABEL.ENTITY2%%%")
call MAKE_DSL_ENTITY2_1()
end ; OCC_ENTITY2
#endif ; undefined OCC_ENTITY2»
```

UMerkel_Babysteps3_3.JPG



UMerkel_Babysteps3_4.JPG