Mit 'Babysteps' in die XTEXT-XTEND Welt

Teil 8: Luecken(x)text, jetzt flexibler und mit Referenzen

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

von Ulrich Merkel

Lxt1: Eine flexible Spezifikation von Vorlagenverzeichnis und Dateityp

```
// Grammar file: Lxt1.xtext
grammar de.ulrichmerkel.lueckenxtext.lxt1.Lxt1
with org.eclipse.xtext.common.Terminals
generate lxt1 "http://www.ulrichmerkel.de/lueckenxtext/lxt1/Lxt1"

Model: ('mergedir=' mergedir=STRING)? ('mergeext=' mergeext=ID)?
       modules+=Module+;

Module:       (Mergefile);
Mergefile:    'mergefile' name=ID
       'placeholders=' placeholderPairs+=KeyValuePair
       (',' placeholderPairs+=KeyValuePair)* ;
KeyValuePair:name=ID '=' value=STRING;
```

```
// aus: Lxt1Generator.xtend
class Lxt1Generator implements IGenerator {
var mergedir = "..\\!lueckenxtext_home"
var mergeext = "txt"
def doPreparation(Model sm) {
       mergedir = sm.mergedir?:"..\\!lueckenxtext_home"
       mergeext = sm.mergeext?:"txt"
       return ""
}
```

```
// aus: Lxt1ProposalProvider.xtend
override completeMergefile_Name(EObject model, Assignment assignment,
       ContentAssistContext context, ICompletionProposalAcceptor acceptor)
{
       completeRuleCall(((assignment.getTerminal()as RuleCall)), context,
acceptor);
       // now we get access to the rootContainer, (here it is not necessary
because we are on Model)
       val rootContainer = EcoreUtil2.getRootContainer(model) as Model
//     addFilesFromDir("..\\!lueckenxtext_home","txt",context,acceptor)
       addFilesFromDir(rootContainer.mergedir?:"..\\!lueckenxtext_home",
              rootContainer.mergeext?:"txt",context,acceptor)
}
```

```
// aus: Lxt1QuickfixProvider.xtend, def addMergefileValues

// [ context |
//  getRootContainer() requires an EObject, so element argument is added
[ element, context |
val rootContainer = EcoreUtil2.getRootContainer(element) as Model
val xtextDocument = context.xtextDocument
var replaceText = " placeholders=mergefilename=" + '"' + issue.data.get(0)
+ '"'
var mergefilepath = rootContainer.mergedir?:"..\\!lueckenxtext_home"
mergefilepath += "\\" + issue.data.get(0) + "."
mergefilepath += rootContainer.mergeext?:"txt"
replaceText += makeDefaultPlaceholderGroup(mergefilepath)
xtextDocument.replace(issue.offset + issue.getLength, 0, replaceText)
]
```

```
/*
 * generated by Xtext: Lxt1Generator.xtend
 */
package de.ulrichmerkel.lueckenxtext.lxt1.generator

import de.ulrichmerkel.lueckenxtext.lxt1.lxt1.KeyValuePair
import de.ulrichmerkel.lueckenxtext.lxt1.lxt1.Mergefile
import de.ulrichmerkel.lueckenxtext.lxt1.lxt1.Model
import de.ulrichmerkel.lueckenxtext.lxt1.lxt1.Module
import java.awt.Toolkit
import java.awt.datatransfer.Clipboard
import java.awt.datatransfer.StringSelection
import java.io.File
import java.nio.charset.StandardCharsets
import java.util.HashMap
import java.util.regex.Matcher
import java.util.regex.Pattern
import javax.swing.JOptionPane
import org.eclipse.emf.common.util.EList
import org.eclipse.emf.ecore.resource.Resource
import org.eclipse.xtext.generator.IFileSystemAccess
import org.eclipse.xtext.generator.IGenerator

import static com.google.common.io.Files.*

class Lxt1Generator implements IGenerator {
var mergedir = "..\\!lueckenxtext_home"
var mergeext = "txt"
def doPreparation(Model sm) {
	mergedir = sm.mergedir?:"..\\!lueckenxtext_home"
	mergeext = sm.mergeext?:"txt"
	return ""
}

def addIncludeFile(String filename) '''
«val File incfile = new File(mergedir + "\\" + filename + "." + mergeext)»
«IF incfile.exists()»
«toString(incfile,  StandardCharsets.UTF_8)»
«ELSE»
«System.out.println("Error: addIncludeFile: >"
	+ mergedir + "\\" + filename + "." + mergeext +"< not found")»
«ENDIF»
'''

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

def checkForPlaceholderNames(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), "")
		}
		System.out.println(
			"Warning1: " + hashNames.size + " unresolved placeholders in "
+ fileID + ": " +
				hashNames.keySet.join(", ")
		)
	}
	return ""
}

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
}

def showAlert(String theMessage, String title) {
        JOptionPane.showMessageDialog(null, theMessage, title
                , JOptionPane.INFORMATION_MESSAGE);
        return ""
}


/* now the main thing */
override void doGenerate(Resource resource, IFileSystemAccess fsa) {
        var String scriptFileName = resource.getURI().lastSegment.toString();
        var String outFileBase = scriptFileName.substring(0,
scriptFileName.lastIndexOf("."));
        fsa.generateFile(outFileBase + '.1.CODE',
makeTheCodeAutocopy(resource.contents.head as Model))
//      fsa.generateFile(outFileBase + '.1.CODE', makeTheCode(resource.contents.head
as Model))
}

def makeTheCodeAutocopy(Model sm) '''
«setClipboard(makeTheCode(sm).toString)»
'''

def makeTheCode(Model sm) {
        doPreparation(sm)
        makeTheModules(sm)
}

def makeTheModules(Model sm) '''
«FOR Module modul : sm.modules»
«switch modul {
                Mergefile: {addMergeFile(modul)}
        }»
«ENDFOR»
'''

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

}
```

Lxt1 in der Anwendung

```
mergedir="..\\!lueckenxtext_home_user" mergeext=txt1
mergefile test1f_b placeholders=mergefilename="test1f_b"
,moreplaceholder="abc"
,moreplaceholder2="d12"
,moreplaceholder3="xxx"
```

Lxt2: "refentity--", "reffield--" Platzhalter für Referenzen

```
// Grammar file: Lxt2.xtext
grammar de.ulrichmerkel.lueckenxtext.lxt2.Lxt2
with org.eclipse.xtext.common.Terminals
generate lxt2 "http://www.ulrichmerkel.de/lueckenxtext/lxt2/Lxt2"

Model: ('mergedir=' mergedir=STRING)? ('mergeext=' mergeext=ID)?
        entities+= Entity*
        modules+=Module+
;

Module:       (Mergefile);
Mergefile:    'mergefile' name=ID
        'placeholders=' placeholderPairs+=KeyValuePair? (','
placeholderPairs+=KeyValuePair)* ;
KeyValuePair:(KeyValuePair1 | KeyValuePair2 | KeyValuePair3 );
KeyValuePair1:      name=ID '=' value=STRING;
KeyValuePair2:      'reffield--'name=ID '=' reffield=[Field|QualifiedName];
KeyValuePair3:      'refentity--'name=ID '=' refentity=[Entity];
```

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

Entity: 'entity' name=ID '.' model=ID 'fields='
     fields += Field? (',' fields+= Field)* ;
Field: name=ID ;



// aus: Lxt2Generator.xtend
for (KeyValuePair kvp : preferences) {
//mergeTextWork = mergeTextWork.replaceAll("\\$\\{" + kvp.name + "\\}",
kvp.value);
       mergeTextWork = mergeTextWork.replaceAll
              ("\\$\\{" + retKvpRealName(kvp) + "\\}", retKvpValue(kvp));
}

<snip>

// returning the "real" names of the placeholders
def dispatch retKvpRealName(KeyValuePair1 kvp1) { return(kvp1.name) }
def dispatch retKvpRealName(KeyValuePair2 kvp2)
       { return("reffield--"+kvp2.name) }
def dispatch retKvpRealName(KeyValuePair3 kvp3)
       { return("refentity--"+kvp3.name) }

// returning the values for replacement
def dispatch retKvpValue(KeyValuePair1 kvp1) { return(kvp1.value) }
def dispatch retKvpValue(KeyValuePair2 kvp2) {
       return(kvp2.reffield.name + "." +
              genParentname(kvp2.reffield.eContainer()))
}
def dispatch retKvpValue(KeyValuePair3 kvp3)
       { return(kvp3.refentity.name) }

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

In Teil 4 der Artikelreihe hatte ich dispatch noch nicht gekannt und deshalb noch selbst mit einer switch Konstruktion diese Aufgabe mit deutlich mehr Aufwand implementiert; man lernt halt doch noch dazu.

```
// aus: Umecode3fGenerator.xtend

// returning the "real" names of the placeholders
       def retKvpRealName(KeyValuePair kvp) {
              switch kvp {
                     KeyValuePair1: { return(retKvp1RealName(kvp)) }
                     KeyValuePair2: { return(retKvp2RealName(kvp)) }
                     KeyValuePair3: { return(retKvp3RealName(kvp)) }
              }
       }

       def retKvp1RealName(KeyValuePair1 kvp1) { return(kvp1.name) }
       def retKvp2RealName(KeyValuePair2 kvp2)
{ return("reffield:"+kvp2.name) }
       def retKvp3RealName(KeyValuePair3 kvp3)
{ return("refentity:"+kvp3.name) }
// returning the values for replacement
       def retKvpValue(KeyValuePair kvp) {
              switch kvp {
                     KeyValuePair1: { return(retKvp1Value(kvp)) }
                     KeyValuePair2: { return(retKvp2Value(kvp)) }
                     KeyValuePair3: { return(retKvp3Value(kvp)) }
              }
       }
```

```
	def retKvp1Value(KeyValuePair1 kvp1) { return(kvp1.value) }
	def retKvp2Value(KeyValuePair2 kvp2) {
		return(kvp2.reffield.name + "." +
genParentname(kvp2.reffield.eContainer()))
	}
	def retKvp3Value(KeyValuePair3 kvp3) { return(kvp3.refentity.name) }

	def checkForPlaceholders(String workText, String fileID) {
		checkForPlaceholderNames(workText, fileID)
		return ""
	}
```

Lxt2 in der Anwendung

```
entity a2.test2 fields=A2, A5
entity ff.model1 fields=A2

mergefile b_ref placeholders=mergefilename="b_ref"
,moreplaceholder=""
,moreplaceholder1=""
,refentity--aentity=
,reffield--afield=
```

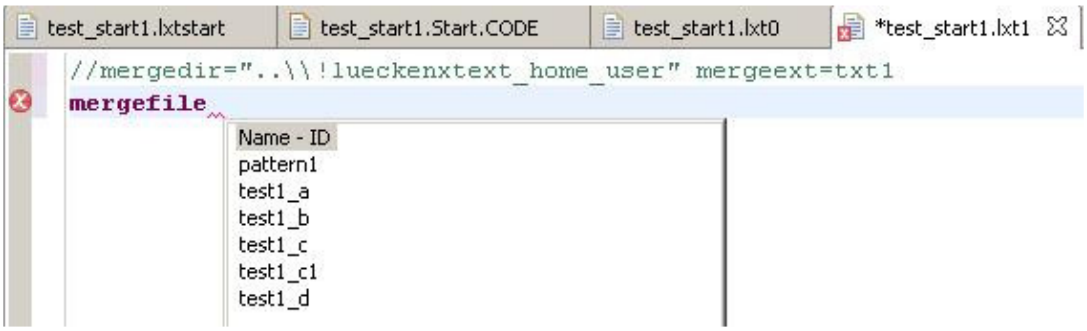
Links & Literatur

[1] Luna DSL Package download https://www.eclipse.org/downloads/packages/

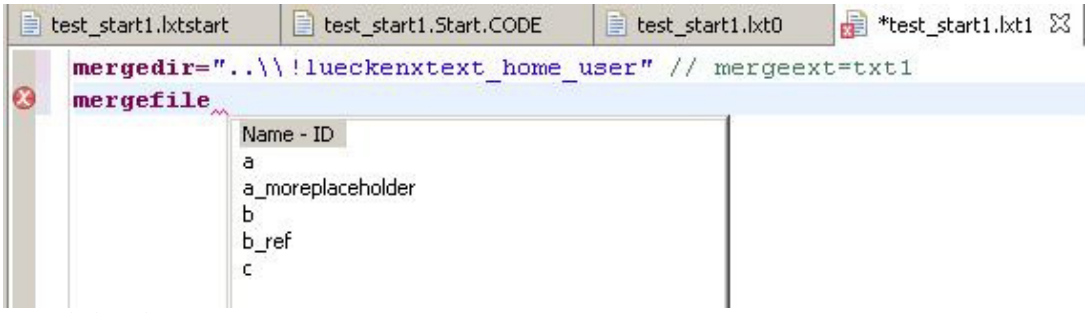
[2] Material zum Mitmachen: !lueckenxtext_luna-part2-all-in-one.zip auf www.uli-merkel.de/uli-xtext

[3] Coding Nights auf www.codingnights.co

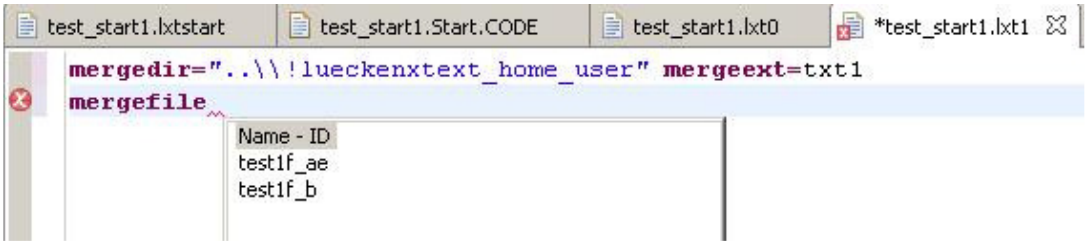
BILDÜBERSICHT



UMerkel_Babysteps8_1.JPG

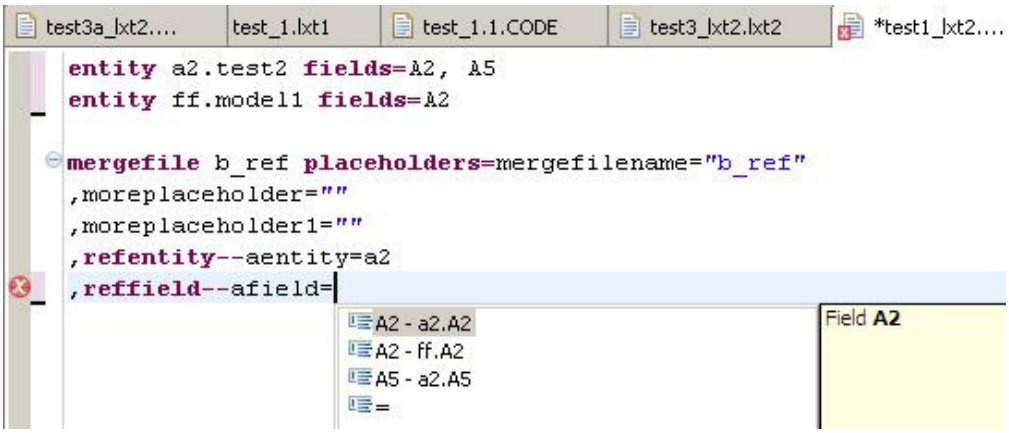


UMerkel_Babysteps8_2.JPG

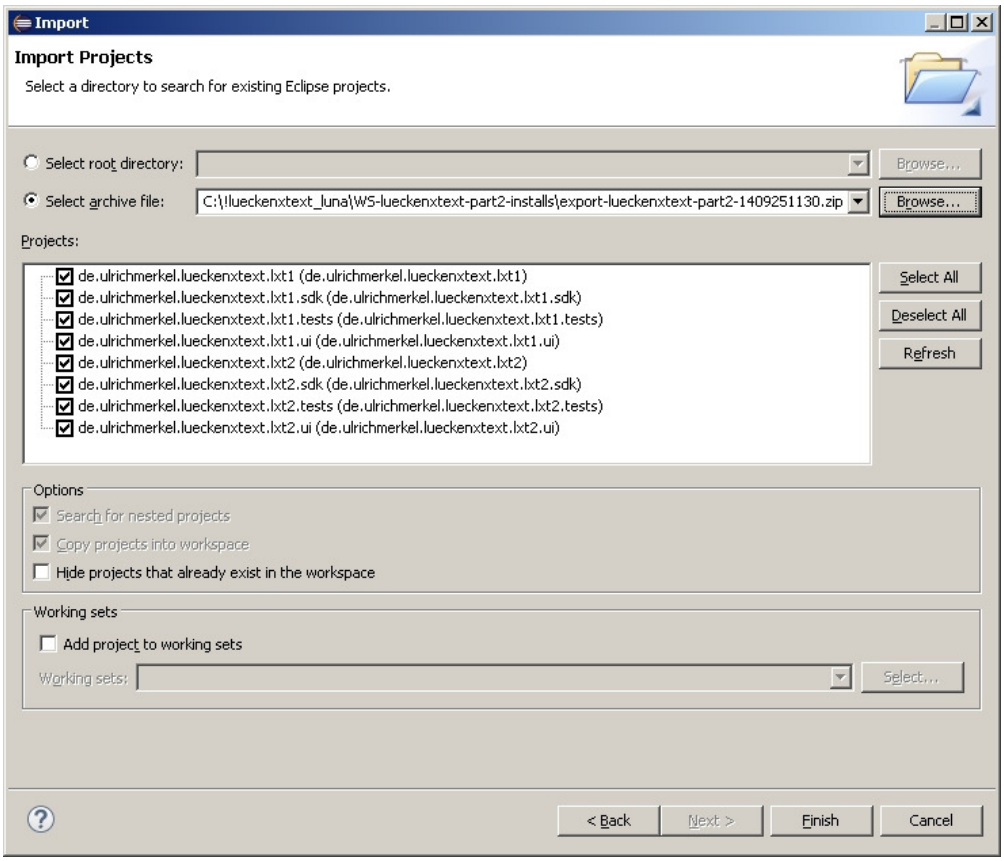


UMerkel_Babysteps8_3.JPG

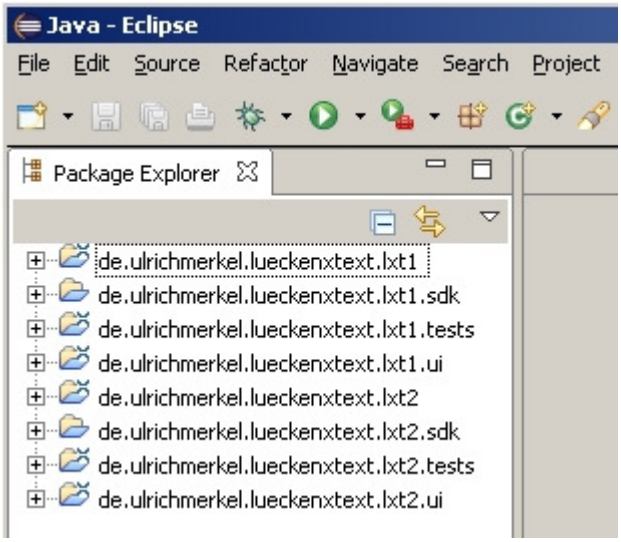
UMerkel_Babysteps8_4.JPG



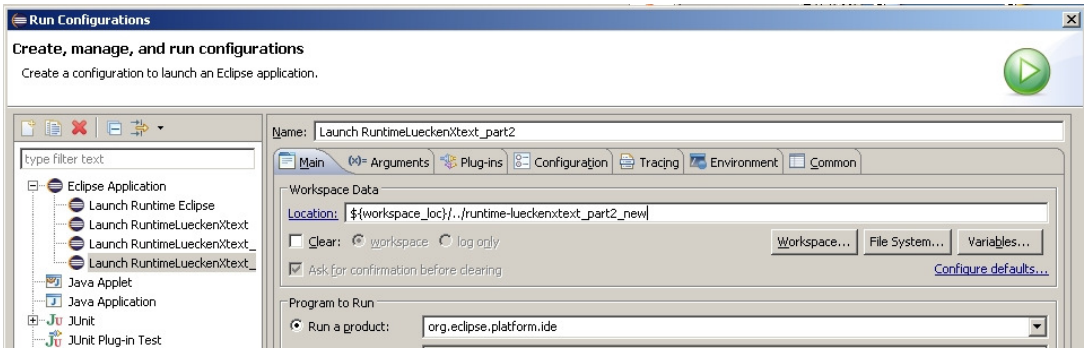
UMerkel_Babysteps8_5.JPG



UMerkel_Babysteps8_10.JPG



UMerkel_Babysteps8_11.JPG



UMerkel_Babysteps8_12.JPG