Mit 'Babysteps' in die XTEXT-XTEND Welt

Teil 4: externer Codegenerator mit Umgebungskenntnis dank DSL

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

von Ulrich Merkel

Erweiterung der Grammatik zur Festlegung des Datenmodells

```
// ##### Ausschnitt aus der Grammatik (XTEXT)
// ###############
Model:
      entities+= Entity*
      modules+=Module*
;
Entity:
      'entity' name=ID '.' model=ID
      'fields=' fields += Field? (',' fields+= Field)*
;
Field:
      name=ID
;
```

Nutzung der Datenmodell Informationen mittels interne Referenzen

```
// ##### Ausschnitt aus der Grammatik (*.XTEXT)
// ###############
EntitySupportFile:  'entitysupportfile' entity=ID 'action='
action=('OCC'|'LOOP');

// ##### Ausschnitt aus einer DSL (*.umecode3)
// ###############
entitysupportfile abc action=OCC

// ##### Ausschnitt aus dem Generator (*.XTEND)
// ###############
def genEntitySupportFile(EntitySupportFile module) {
var mergeTextWork = addIncludeFile("umecode1_action_entity.txt").toString;
if (mergeTextWork != null) {
      mergeTextWork =
mergeTextWork.replaceAll("\\$\\{actioncode\\}",module.action.toUpperCase);
      mergeTextWork =
mergeTextWork.replaceAll("\\$\\{entityname\\}",module.entity.toUpperCase);
      if (mergeTextWork.contains("${")) {
      addErr("Still unresolved placeholders in: >" +mergeTextWork+"<")
      }
    return mergeTextWork }
else return ""
}
```

Es wird jetzt eine Erweiterung erstellt, in der "entityname" als Referenz genutzt wird:

```
// ##### Ausschnitt aus der Grammatik (*.XTEXT)
// ###############
EntitySupportCross:  'entitysupportcross' entity=[Entity] 'action='
action=('OCC'|'LOOP');

// ##### Ausschnitt aus einer DSL (*.umecode3)
// ###############
entitysupportfile abc action=OCC
```

```
// ##### Ausschnitt aus dem Generator (*.XTEND)
// ###############
def genEntitySupportCross(EntitySupportCross module) {
var mergeTextWork = addIncludeFile("umecode1_action_entity.txt").toString;
if (mergeTextWork != null) {
      mergeTextWork = mergeTextWork.replaceAll("\\$\\{actioncode\\}",
            module.action.toUpperCase);
      mergeTextWork = mergeTextWork.replaceAll("\\$\\{entityname\\}",
            module.entity.name.toUpperCase);
      checkForPlaceholders(mergeTextWork, "umecode1_action_entity.txt")
      return mergeTextWork
} else
      return ""
}
```

Varianten der Ersetzungspaare in der Grammatik

Da eine ID keinen Doppelpunkt enthalten darf, müssen wir die Ersetzungspaare erweitern:

```
// ##### Ausschnitt aus der Grammatik (*.XTEXT)
// ###############
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)*;
```

Varianten der Ersetzungspaare im Generator

```
// ##### Ausschnitt aus dem Generator (*.XTEND)
// ###############
for (KeyValuePair kvp : kvps) {
      mergeTextWork = mergeTextWork.replaceAll("\\$\\{"
            + retKvpRealName(kvp) + "\\}", retKvpValue(kvp));
}
// 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 genParentname(EObject container) '''«switch container {
      Entity: {
            container.name
            }
      }»
'''
```

Ein erster Test des Vorgehens

```
// ##### Ausschnitt aus der Grammatik (*.XTEXT)
// ###############
KvpTest:    'kvptest=' values+=KeyValuePair (','
values+=KeyValuePair)*;

// ##### Ausschnitt aus dem Generator (*.XTEND)
// ###############
def genKvpTest(KvpTest kvpt) {'''
      Result of the kvptest= «»
      «FOR KeyValuePair kvpv : kvpt.values»
      -  «retKvpRealName(kvpv)»  --  «retKvpValue(kvpv)»
      «ENDFOR»
      '''
      }

// ##### Ausschnitt aus einer DSL (*.umecode3)
// ###############
kvptest=abc="Value", reffield:test12=UXGROUP.BREAK,
refentity:ab=UXFIELD

// ##### Ausschnitt aus der generierten Datei
// ###############
Result of the kvptest=
-  abc  -- Value
-  reffield:test12  -- BREAK.UXGROUP
-  refentity:ab  -- UXFIELD
```

Komfortabel und noch schneller: Optionales Generieren in die Zwischenablage

```
// ##### Ausschnitt aus dem Generator (*.XTEND)
// ###############
fsa.generateFile(outFileBase + '.2F.CODE',
       makeTheCodeAutocopy(resource.contents.head as Model))

def makeTheCodeAutocopy(Model sm) '''
«loadMyOptions(sm.options)»
«IF isMyOption("autocopy")»
      «setClipboard(makeTheCode(sm).toString)»
      «showAlert("autocopy activated\n"+reportLog().toString)»
«ELSE»
«makeTheCode(sm).toString»
«ENDIF»
'''
```

"The Hitchhiker's Guide to XTEXT-XTEND": myOptions steuern den Generator

```
// ##### Ausschnitt aus der Grammatik (*.XTEXT)
// ###############
Model:
    ('generator_options' options+=Option? (',' options+=Option)*)*
.....
Option:     name=ID ('=' value=STRING)?;

// ##### Ausschnitt aus dem Generator (*.XTEND)
```

```
// ###############
// *** 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))
}

// *** die Einbindung erfolgt durch
def universalPreparation(Model sm) {
      loadMyOptions(sm.options)
      return ""
}

/* now the main thing */
override doGenerate(Resource resource, IFileSystemAccess fsa) {
var String scriptFileName =
      resource.getURI().lastSegment.toString();
scriptfileName = scriptFileName;
var String outFileBase =
      scriptFileName.substring(0, scriptFileName.lastIndexOf("."));
universalPreparation(resource.contents.head as Model)
resetLog()
...
```

"The Hitchhiker's Guide to XTEXT-XTEND": direkt in die Zwischenablage

```
// ##### Ausschnitt aus dem Generator (*.XTEND)
// ###############
import java.awt.Toolkit
import java.awt.datatransfer.Clipboard
import java.awt.datatransfer.StringSelection
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 makeTheCodeAutocopy(Model sm) '''
«setClipboard(makeTheCode(sm).toString)»
'''
```

```
// **** Möglichkeit, im Dialog Meldungen angeben zu können
import javax.swing.JOptionPane
def showAlert(String theMessage) {
      JOptionPane.showMessageDialog(null, theMessage,
            scriptfileName, JOptionPane.INFORMATION_MESSAGE);
      return ""
}
```

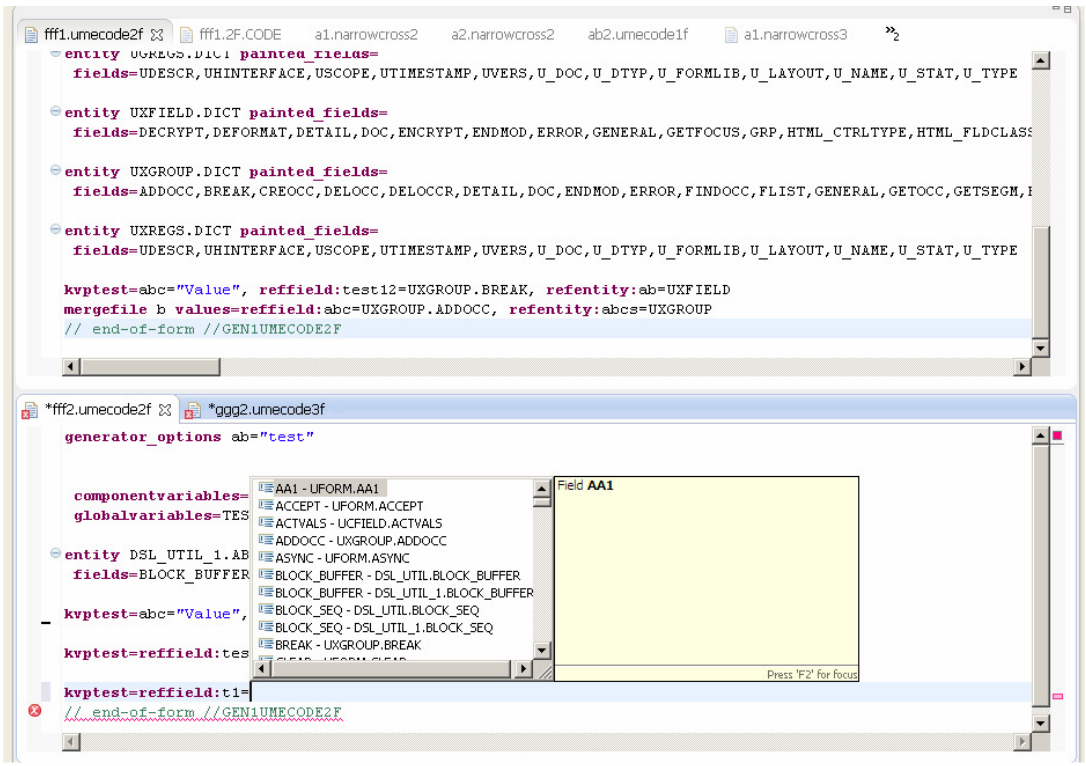
"The Hitchhiker's Guide to XTEXT-XTEND": Scoping nur innerhalb der DSL Datei

```
// ##### Ausschnitt aus der Grammatik (MyDsl.xtext)
// ###############
grammar org.xtext.example.mydsl.MyDsl with
org.eclipse.xtext.common.Terminals
generate myDsl "http://www.xtext.org/example/mydsl/MyDsl"
Model:
      greetings+=Greeting*
      farewells+=Farewell*;
Greeting:
      'Hello' name=ID '!';
Farewell:
      'Bye' greeter=[Greeting];
```

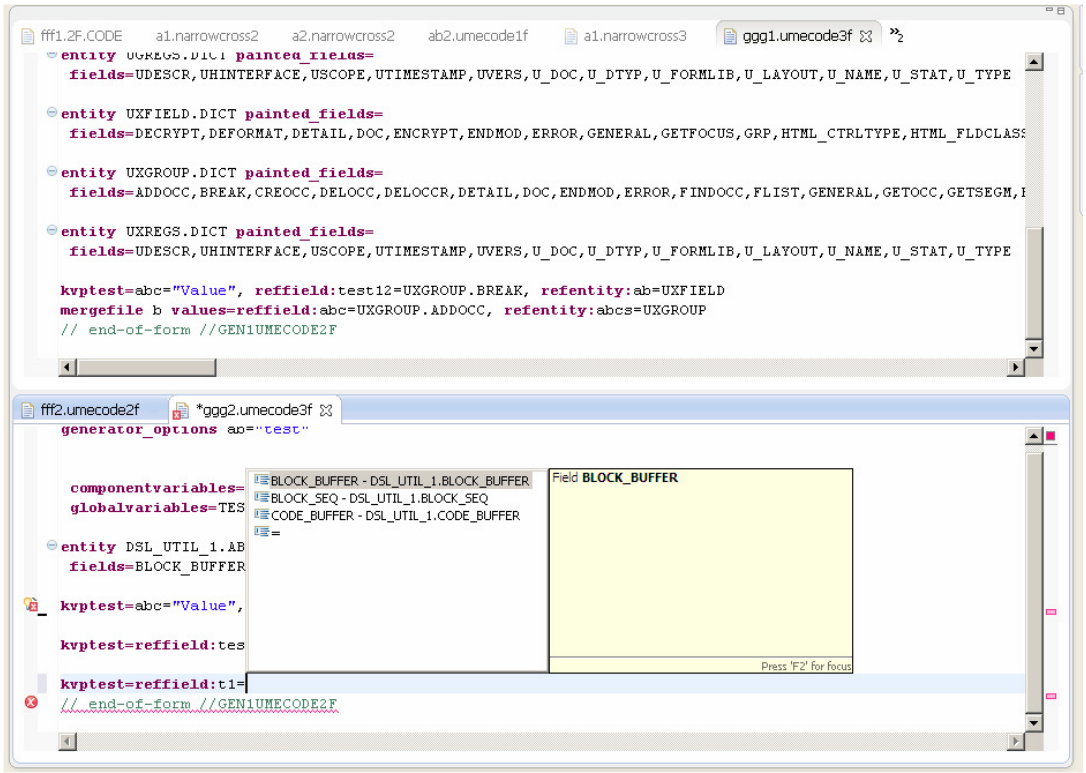
Im Verzeichnis org.xtext.example.mydsl/src/ org.xtext.example.mydsl wird für das Interface IGlobalScopeProvider die Klasse **NullGlobalScopeProvider.java** angelegt. Im gleichen Verzeichnis liegt die Datei **MyDslRuntimeModule.java**, indem wir die eigentliche Implementierung überschreiben, damit zur Laufzeit das geänderte Scoping benutzt wird:

```
// ##### unser neuer ScopeProvider (NullGlobalScopeProvider.java)
// ###############
package org.xtext.example.mydsl;
import org.eclipse.emf.ecore.EReference;
import org.eclipse.emf.ecore.resource.Resource;
import org.eclipse.xtext.resource.IEObjectDescription;
import org.eclipse.xtext.scoping.IGlobalScopeProvider;
import org.eclipse.xtext.scoping.IScope;
import com.google.common.base.Predicate;
public class NullGlobalScopeProvider implements
IGlobalScopeProvider {
@Override
public IScope getScope(Resource context, EReference reference,
            Predicate<IEObjectDescription> filter) {
      return IScope.NULLSCOPE;
}
}
// ##### Einbindung in MyDslRuntimeModule.java
// ###############
package org.xtext.example.mydsl;
import org.eclipse.xtext.scoping.IGlobalScopeProvider;
public class MyDslRuntimeModule extends
org.xtext.example.mydsl.AbstractMyDslRuntimeModule {
      @Override
      public Class<? extends IGlobalScopeProvider>
bindIGlobalScopeProvider() {
            return NullGlobalScopeProvider.class;
      }
}
```

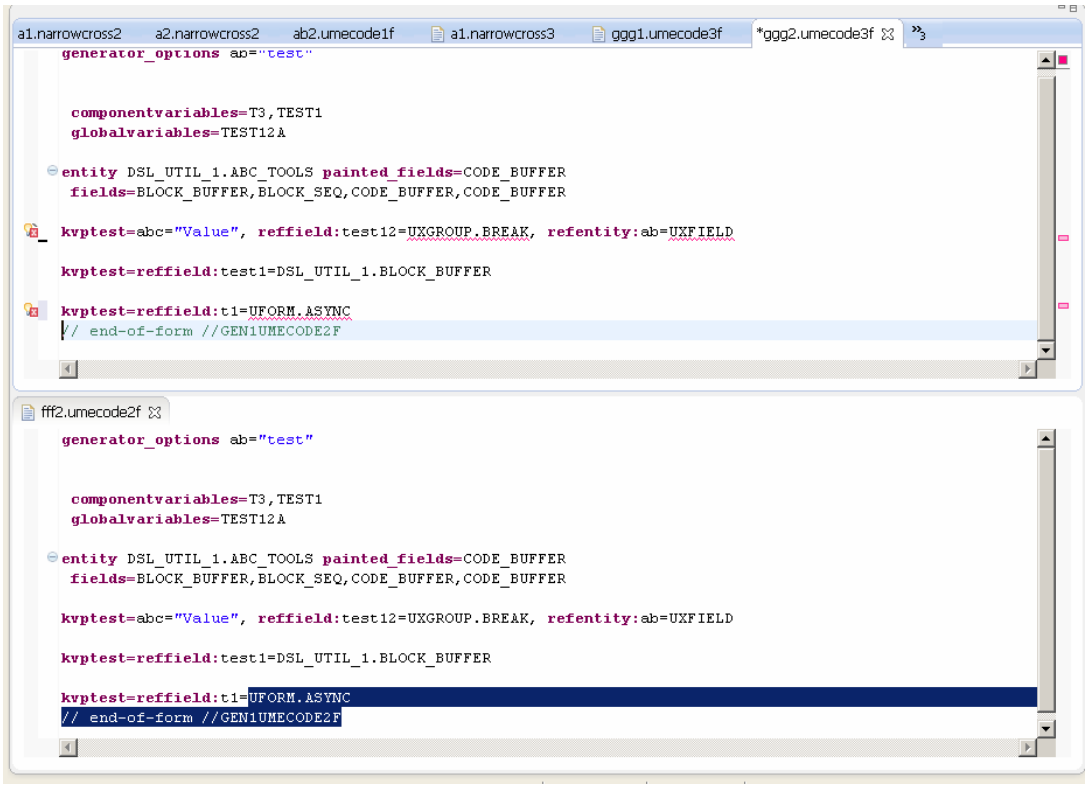
BILDÜBERSICHT



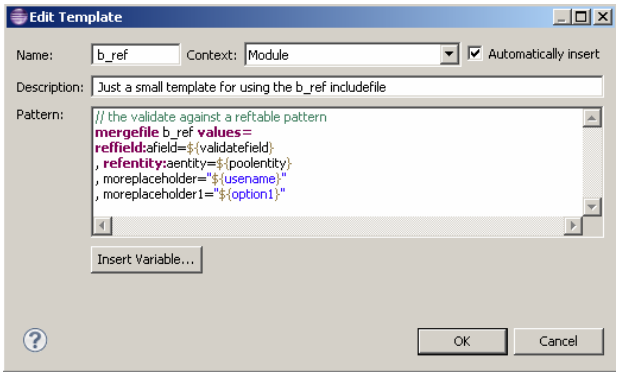
UMerkel_Babysteps4_1.JPG



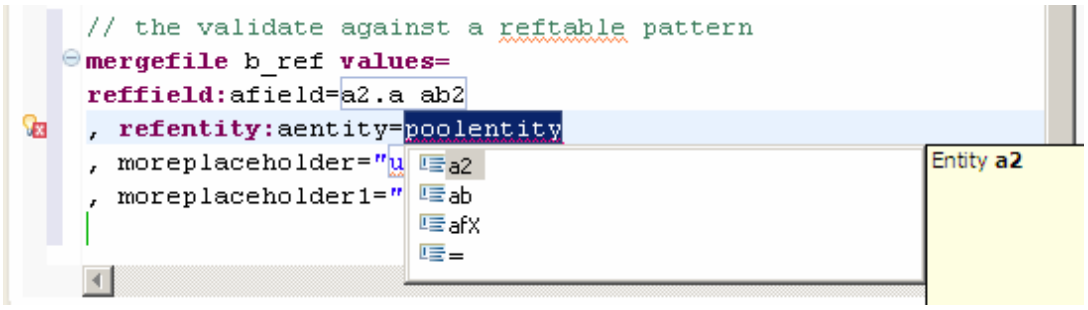
UMerkel_Babysteps4_2.JPG



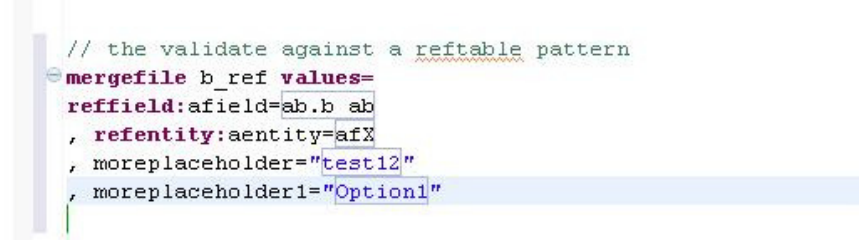
UMerkel_Babysteps4_3.JPG



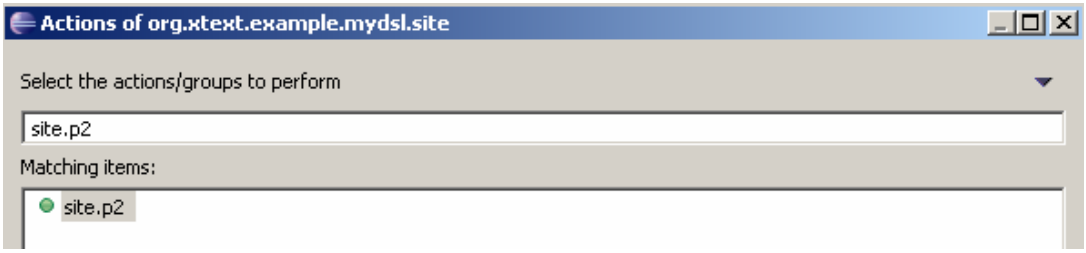
UMerkel_Babysteps4_4.JPG



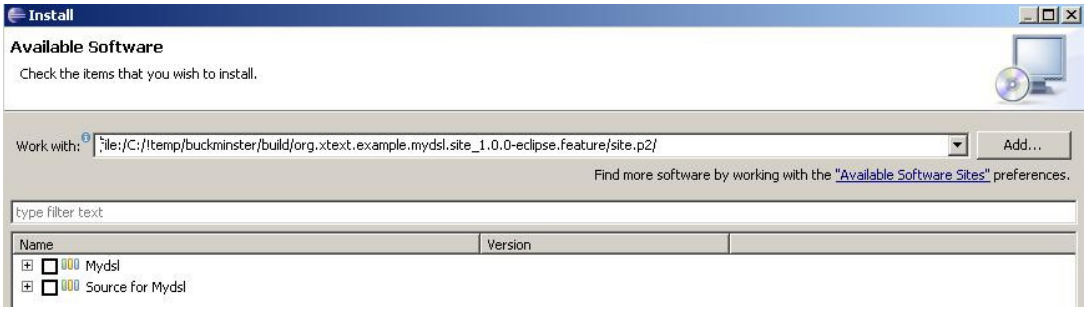
UMerkel_Babysteps4_5.JPG



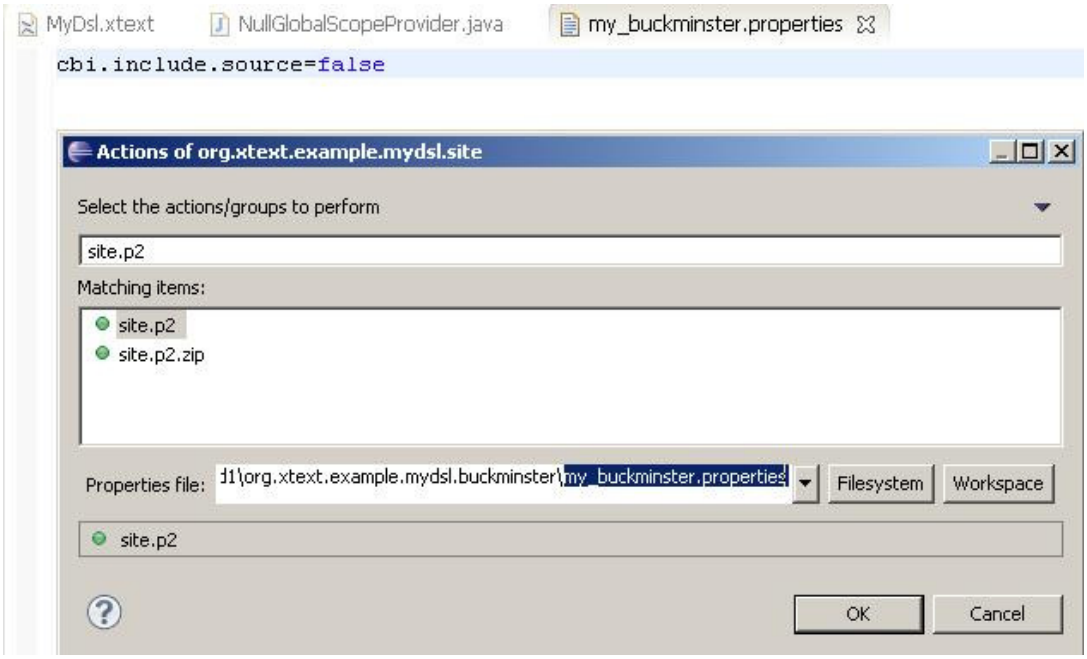
UMerkel_Babysteps4_6.JPG



UMerkel_Babysteps4_7.JPG



UMerkel_Babysteps4_8.JPG



UMerkel_Babysteps4_9.JPG