Mit 'Babysteps' in die XTEXT-XTEND Welt

Teil 6: schnelle Visualisierung von Abhängigkeiten 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

Textblöcke werden im Generator intern in Cartridges verwaltet

```
def addInternFileYed(String selection) '''
«IF selection.equalsIgnoreCase("visout_first")»
      <?xml version="1.0" encoding="UTF-8" standalone="no"?>
      <graphml
xmlns="http://graphml.graphdrawing.org/xmlns/graphml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:y="http://www.yworks.com/xml/graphml"
xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns/graphml
http://www.yworks.com/xml/schema/graphml/1.0/ygraphml.xsd">
        <key for="node" id="d0" yfiles.type="nodegraphics"/>
        <key attr.name="description" attr.type="string" for="node"
id="d1"/>
        <key for="edge" id="d2" yfiles.type="edgegraphics"/>
        <graph edgedefault="directed" id="G">
«ELSEIF selection.equalsIgnoreCase("visout_last")»
        </graph>
      </graphml>
«ELSEIF selection.equalsIgnoreCase("visout_class")»
    <node id="${entityname}"><data key="d0"><y:UMLClassNode>
          <y:Geometry height="25.0" width="200.0"/>
          <y:NodeLabel>${entityname}</y:NodeLabel>
        </y:UMLClassNode></data>
      <data key="d1">${entitydescription}</data>
    </node>
«ELSEIF selection.equalsIgnoreCase("visout_assoc")»
    <edge id="${entityfrom}-${entityto}" source="${entityfrom}"
target="${entityto}">
      <data key="d2"><y:PolyLineEdge><y:Arrows
target="standard"/></y:PolyLineEdge></data>
    </edge>
«ELSEIF selection.equalsIgnoreCase("visout_assoc_bold")»
    <edge id="${entityfrom}-${entityto}" source="${entityfrom}"
target="${entityto}">
      <data key="d2"><y:PolyLineEdge><y:Arrows
target="diamond"/></y:PolyLineEdge></data>
    </edge>
«ELSEIF selection.equalsIgnoreCase("visout_ext")».graphml
«ELSE»
«addLog("Error: addInternFile: File >" + selection + "< not
implemented")»
«ENDIF»
'''
```

Bedeutend einfacher sind die Vorgaben für eine DOT Datei, da hier keine Entitätsangaben benötigt werden.

```
// output pattern for graphviz DOT file
def addInternFileDot(String selection) '''
«IF selection.equalsIgnoreCase("visout_first")»
      digraph ${outfileid} {
«ELSEIF selection.equalsIgnoreCase("visout_last")»
      }
```

```
«ELSEIF selection.equalsIgnoreCase("visout_class")»
«ELSEIF selection.equalsIgnoreCase("visout_assoc")»
    "${entityfrom}" -> "${entityto}"
«ELSEIF selection.equalsIgnoreCase("visout_assoc_bold")»
    "${entityfrom}" -> "${entityto}" [arrowhead = diamond]
«ELSEIF selection.equalsIgnoreCase("visout_ext")».udot
«ELSE»
«addLog("Error: addInternFile: File >" + selection + "< not
implemented")»
«ENDIF»
'''
```

Unterschiedliche Ausgabeoptionen

```
class Umevis0Generator implements IGenerator {
      var outputVersion = "" // yed, Xmi, dot



def addInternFile(String selection) '''
«IF outputVersion.equalsIgnoreCase("yed")»
«addInternFileYed(selection)»
«ELSEIF outputVersion.equalsIgnoreCase("xmi")»
«addInternFileXmi(selection)»
«ELSEIF outputVersion.equalsIgnoreCase("dot")»
«addInternFileDot(selection)»
«ELSE»
«addInternFileYed(selection)»
«ENDIF»
'''
```

Die Teilbausteine und ihre Verwendung

```
def genVisout1(String outfileid, String outfiledescription) {
var mergeTextWork = addInternFile("visout_first").toString;
if (mergeTextWork != null) {
      mergeTextWork =
mergeTextWork.replaceAll("\\$\\{outfileid\\}",
outfileid.toUpperCase.escapeStringToXml);
      mergeTextWork =
mergeTextWork.replaceAll("\\$\\{outfiledescription\\}",
            outfiledescription.escapeStringToXml);
      return mergeTextWork
} else
      return ""
}

def genVisout2(String entityname, String entitydescription) {
var mergeTextWork = addInternFile("visout_class").toString;
if (mergeTextWork != null) {
mergeTextWork = mergeTextWork.replaceAll("\\$\\{entityname\\}",
entityname.toUpperCase.escapeStringToXml);
      mergeTextWork =
mergeTextWork.replaceAll("\\$\\{entitydescription\\}",
            entitydescription.escapeStringToXml);
      return mergeTextWork
} else
      return ""
}

def genVisout3(String entityfrom, String entityto, Boolean isBold)
{
```

```
var mergeTextWork = addInternFile("visout_assoc").toString;
if (isBold?:false) mergeTextWork =
addInternFile("visout_assoc_bold").toString;
if (mergeTextWork != null) {
      mergeTextWork =
mergeTextWork.replaceAll("\\$\\{entityfrom\\}",
entityfrom.toUpperCase.escapeStringToXml);
      mergeTextWork = mergeTextWork.replaceAll("\\$\\{entityto\\}",
entityto.toUpperCase.escapeStringToXml);
      return mergeTextWork
} else
      return ""
}

def genVisout4() {
var mergeTextWork = addInternFile("visout_last").toString;
if (mergeTextWork != null) {
      return mergeTextWork
} else
      return ""
}
```

Die Hilfsroutine escapeStringToXml Routine berücksichtigt auch die Ausgabevariante:

```
def String escapeStringToXml(String string) {
// dot is not XML based, so we do not have to escape
      if(outputVersion.equalsIgnoreCase("dot")) {return string}
      if(string != null) string.replaceAll("&", "&amp;").
            replaceAll("<", "&lt;").replaceAll(">", "&gt;").
            replaceAll("\"", "&quot;").replaceAll("'", "&apos;")
}
```

Umevis0: Schnittstellendokumentation und Entwicklung der Textbausteine

```
grammar de.ulrichmerkel.umevis0.Umevis0 with
org.eclipse.xtext.common.Terminals
generate umevis0 "http://www.ulrichmerkel.de/umevis0/Umevis0"
Model: {Model}
('generator_options' options+=Option? (',' options+=Option)*)+;
Option:     name=ID ('=' value=STRING)?;

override void doGenerate(Resource resource, IFileSystemAccess fsa)
{
      var String scriptFileName =
resource.getURI().lastSegment.toString();
      scriptfileName = scriptFileName;
      outFileBase = scriptFileName.substring(0,
scriptFileName.lastIndexOf("."));
      universalPreparation(resource.contents.head as Model)
      resetLog; // resetting the loggging collection
      outputVersion = "yed"
      fsa.generateFile('DEMO.umevis0.graphml',genDemoVisout())
      outputVersion = "xmi"
      fsa.generateFile('DEMO.umevis0.xmi',genDemoVisout())
      outputVersion = "dot"
      fsa.generateFile('DEMO.umevis0.udot',genDemoVisout())
}

def genDemoVisout() {
```

```
	'''
		«genVisout1("OutId", "Outfile description Test12")»
		«genVisout2("Entity1", "testdescription")»
		«genVisout2("Entity2", "anothertest")»
		«genVisout2("Entity3", "a \"test\" => &")»
		«genVisout3("Entity1", "Entity2", true)»
		«genVisout3("Entity1", "Entity3", false)»
		«genVisout3("Entity2", "Entity3", null)»
		«genVisout4()»
	'''
}
```

Umevis1: Einfach nur Caller-Called Paare

```
grammar de.ulrichmerkel.Umevis1 with
org.eclipse.xtext.common.Terminals
generate umevis1 "http://www.ulrichmerkel.de/Umevis1"
Model: {Model}
('generator_options' options+=Option? (',' options+=Option)*)*
callerCalledPairs+=CallerCalledPair*;
Option:     name=ID ('=' value=STRING)?;
CallerCalledPair: name=ID calls=ID (emphasis?='bold')? ;
```

Umevis2: Und jetzt werden die Entitäten auch noch beschrieben

```
grammar de.ulrichmerkel.umevis2.Umevis2 with
org.eclipse.xtext.common.Terminals
generate umevis2 "http://www.ulrichmerkel.de/umevis2/Umevis2"
Model: {Model}
('generator_options' options+=Option? (',' options+=Option)*)*
entities+=Entity*
callerCalledPairs+=CallerCalledPair*;
Option:     name=ID ('=' value=STRING)?;
Entity: name=ID '-' (description=STRING)?;
CallerCalledPair: name=ID calls=ID (emphasis?='bold')? ;
```

Minimaldateien und deren Abbildung

Die DSL Datei minimalst.umevis2 ist der Ausgangspunkt für die folgenden Dateien:

```
abc - "Description of entity abc"
abc def bold
abc def
```

Graphviz (DOT):

```
digraph OUTID1 {
"ABC" -> "DEF" [arrowhead = diamond]
"ABC" -> "DEF"
}
```

yEd (GRAPHML):

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns/graphml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xmlns:y="http://www.yworks.com/xml/graphml"
xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns/graphml
http://www.yworks.com/xml/schema/graphml/1.0/ygraphml.xsd">
  <key for="node" id="d0" yfiles.type="nodegraphics"/>
  <key attr.name="description" attr.type="string" for="node"
id="d1"/>
  <key for="edge" id="d2" yfiles.type="edgegraphics"/>
  <graph edgedefault="directed" id="G">
<node id="ABC"><data key="d0"><y:UMLClassNode>
      <y:Geometry height="25.0" width="200.0"/>
      <y:NodeLabel>ABC</y:NodeLabel>
    </y:UMLClassNode></data>
  <data key="d1">Description of entity abc</data>
</node>
<node id="DEF"><data key="d0"><y:UMLClassNode>
      <y:Geometry height="25.0" width="200.0"/>
      <y:NodeLabel>DEF</y:NodeLabel>
    </y:UMLClassNode></data>
  <data key="d1">-def-</data>
</node>
<edge id="ABC-DEF" source="ABC" target="DEF">
  <data key="d2"><y:PolyLineEdge><y:Arrows
target="diamond"/></y:PolyLineEdge></data>
</edge>
<edge id="ABC-DEF" source="ABC" target="DEF">
  <data key="d2"><y:PolyLineEdge><y:Arrows
target="standard"/></y:PolyLineEdge></data>
</edge>
  </graph>
</graphml>
```

Visual Paradigm for UML (XMI):

```
<?xml version='1.0' ?>
<XMI xmi.version='1.2' xmlns:UML='org.omg.xmi.namespace.UML'>
<XMI.content>
<UML:Model xmi.id='OUTID1' name='Outfile description Test12'
visibility='public'>
<UML:Namespace.ownedElement>
<UML:Class xmi.id='ABC' name='ABC'>
<UML:Classifier.feature><UML:Attribute xmi.id='_-ABC' name='_'
type=''><UML:Attribute.initialValue>
<UML:Expression language='java' body='Description of entity abc'/>
</UML:Attribute.initialValue></UML:Attribute></UML:Classifier.featu
re>
</UML:Class>
```

```
<UML:Class xmi.id='DEF' name='DEF'>
<UML:Classifier.feature><UML:Attribute xmi.id='_-DEF' name='_'
type=''><UML:Attribute.initialValue>
<UML:Expression language='java' body='-def-'/>
</UML:Attribute.initialValue></UML:Attribute></UML:Classifier.featu
re>
</UML:Class>
<UML:Association xmi.id='ABC-DEF'> <UML:Association.connection>
<UML:AssociationEnd xmi.id='ABC-DEF-
1'><UML:AssociationEnd.participant>
<UML:Classifier
xmi.idref='ABC'/></UML:AssociationEnd.participant></UML:Association
End>
<UML:AssociationEnd xmi.id='ABC-DEF-n' aggregation='composite'
><UML:AssociationEnd.participant>
<UML:Classifier
xmi.idref='DEF'/></UML:AssociationEnd.participant></UML:Association
End>
 </UML:Association.connection> </UML:Association>
<UML:Association xmi.id='ABC-DEF'> <UML:Association.connection>
<UML:AssociationEnd xmi.id='ABC-DEF-
1'><UML:AssociationEnd.participant>
<UML:Classifier
xmi.idref='ABC'/></UML:AssociationEnd.participant></UML:Association
End>
<UML:AssociationEnd xmi.id='ABC-DEF-n' aggregation='Aggregation'
><UML:AssociationEnd.participant>
<UML:Classifier
xmi.idref='DEF'/></UML:AssociationEnd.participant></UML:Association
End>
 </UML:Association.connection> </UML:Association>
</UML:Namespace.ownedElement>
</UML:Model>
</XMI.content>
</XMI>
```

Ende

(Default-) Entities aus Beziehungspaaren gewinnen

```
static Map<String, String> theEntities = new TreeMap<String,
String>();

def putTheEntities(String key, String value) {
      theEntities.put(key, value ?: "-"+key+"-")
      return ("")
}
```

```
def getTheEntities(String key) {
	return theEntities.get(key) ?: ""
}

def loadTheEntities(Model sm) {
	theEntities.clear
	for (CallerCalledPair ccp : sm.callerCalledPairs) {
		putTheEntities(ccp.name, null)
		putTheEntities(ccp.calls, null)
	}
	for (Entity entity : sm.entities) {
		putTheEntities(entity.name, entity.description)
	}
	return ""
}
```

Für die Erstellung der Ausgabedateien wird dann die folgende Routine benutzt:

```
def genJobVisout(Model sm) {
	'''
		«genVisout1(outFileBase, "generated by Umevis2")»
		«FOR String key : theEntities.keySet»
		«genVisout2(key, getTheEntities(key))»
		«ENDFOR»
		«FOR CallerCalledPair ccp : sm.callerCalledPairs»
		«genVisout3(ccp.name, ccp.calls, ccp.emphasis)»
		«ENDFOR»
		«genVisout4()»
	'''
}
```

Ende

Babysteps zum Mitmachen

Da wir jetzt eher allgemeine Themen bearbeiten, kann auch mit experimentiert werden. Dazu laden wir uns von www.uli-merkel.de/uli-xtext die Zip Dateien umevisX.zip bzw. runtime-umevisX.zip.

Grundlage ist eine "Full Eclipse" Installation mit Xtext 2.6. Die gepackte Datei kann aus dem Internet von https://www.eclipse.org/Xtext/download.html herunter geladen werden.

Für unsere Beschreibung (auf meinem Windows System) nehmen wir an, dass die ZIP Datei im Verzeichnis C:\!ecl-xtext-v26 entpackt wurde; wir dort also ein Verzeichnis "eclipse" haben.

Beim Starten von Eclipse wird nach dem Pfad zu einem Workspace gefragt, ich gebe hier immer einen relativen Pfad an, also:
`..\WS-umevis`

Jetzt können wir die vorbereiteten Projekte importieren über das Menü:
`File => Import => General => Existing Projects into Workspace`

Im darauf folgenden Bildschirm geben wir den Pfad zu unserem Archiv umevisX.... an und wählen alle enthaltenen Projekte aus.

Als Erstes legen wir eine Run Configuration "umevis" an, die den Workspace "runtime-umevis" nutzt.
`Run => Run Configurations ...`

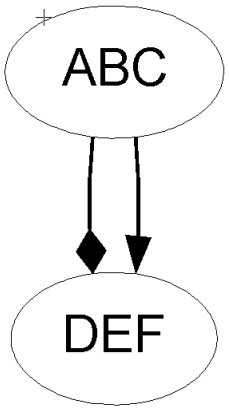
Dabei reicht es aus, wenn wir den Namen in "umevis" ändern und "Finish" drücken.

Wenn wir jetzt "Run Configuration" aufrufen, wird dieser Workspace angelegt und wir kommen in ein neues Eclipse. Hier importieren wir die Testprojekte aus unserem Archiv runtime-umevisX-... .

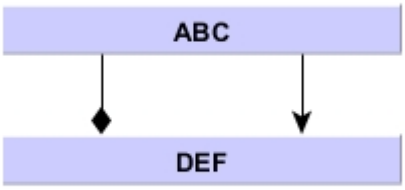
Geschafft!! Unser Experimentalbaukasten samt Testumgebung ist fertig und wir können unsere eigenen Erfahrungen machen.

Ende

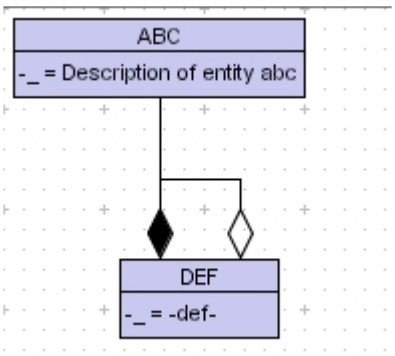
BILDÜBERSICHT



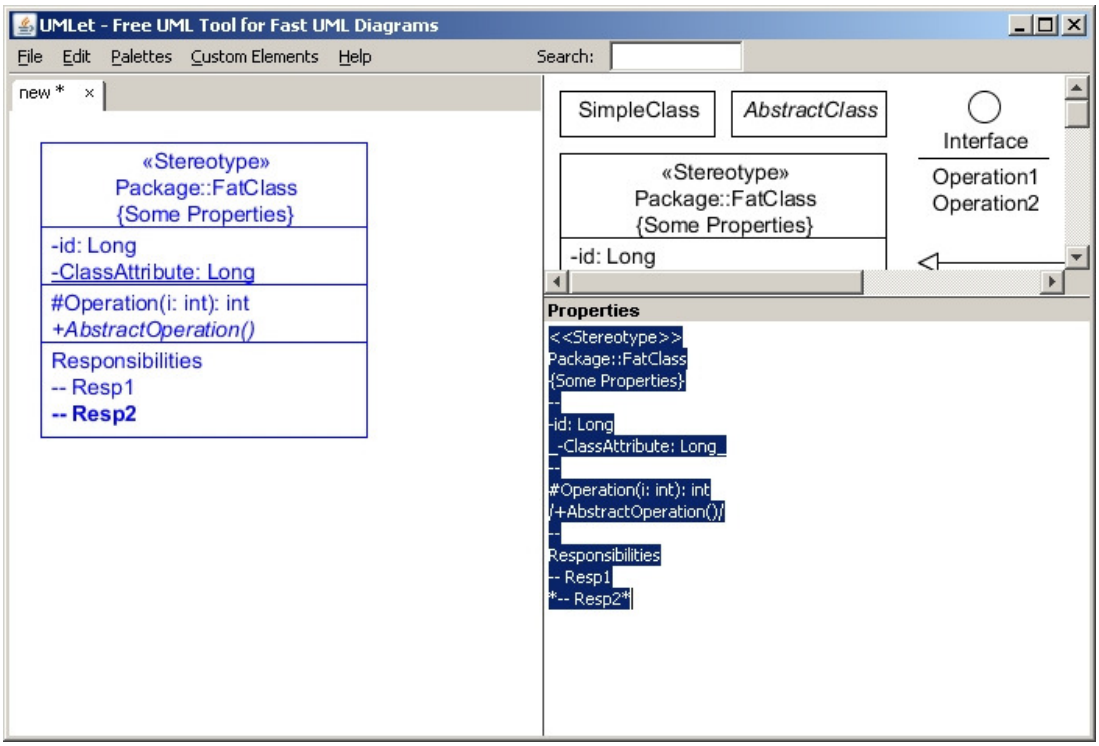
UMerkel_Babysteps6_1.JPG



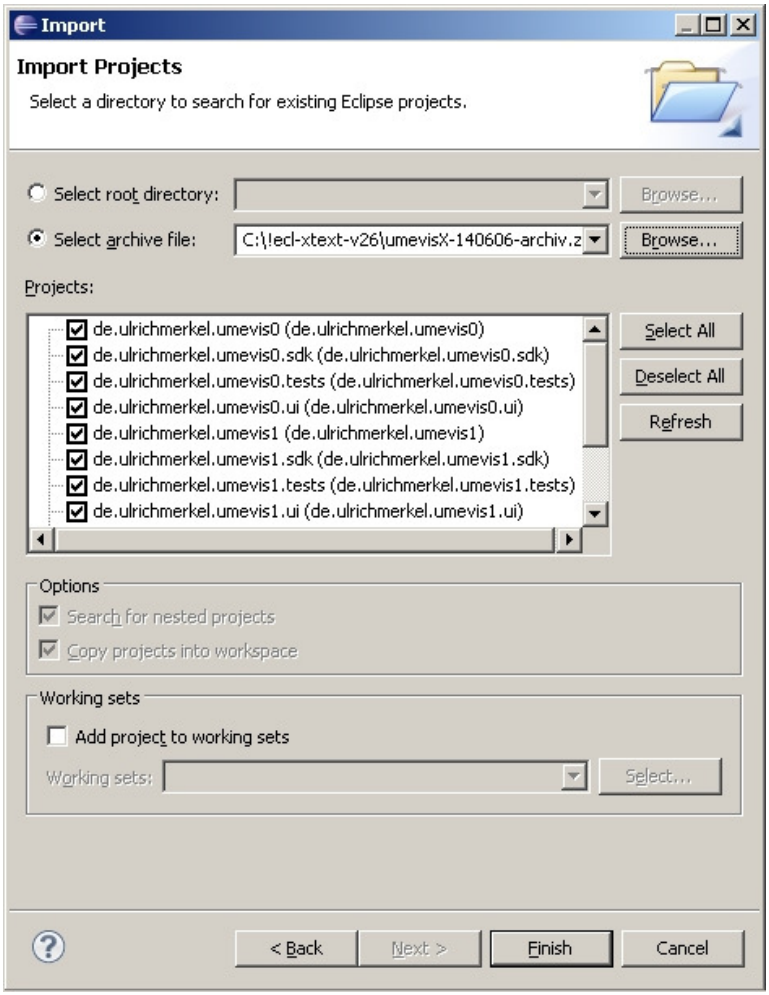
UMerkel_Babysteps6_2.JPG



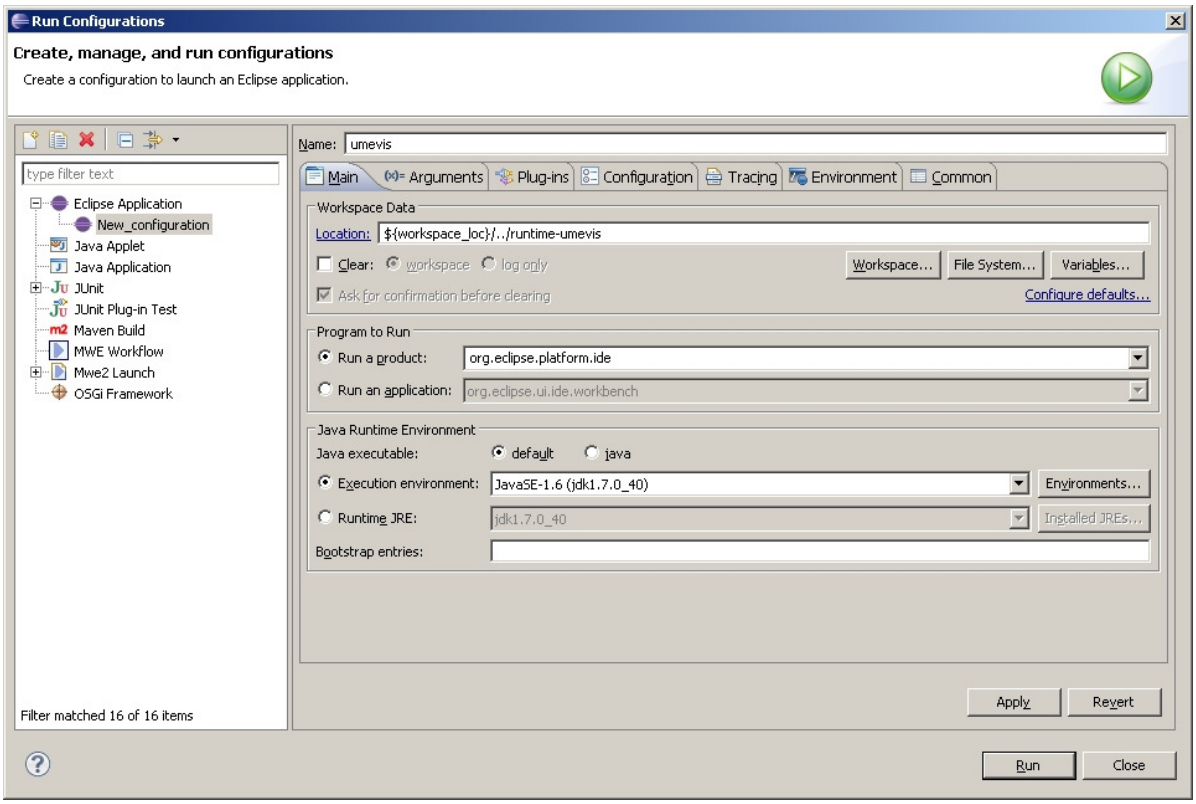
UMerkel_Babysteps6_3.JPG



UMerkel_Babysteps6_4.JPG



UMerkel_Babysteps6_5.JPG



UMerkel_Babysteps6_6.JPG