UNIKASSEL VERSITÄT

SOFTWARE ENGINEERING

Software Engineering II

Software Engineering II – Übung 7

Wintersemester 12/13

Fachgebiet Software Engineering

Andreas Scharf

Installation der MWE Plugins

- **Von der Juno Update Site installieren (falls noch nicht vorhanden):**
  - MWE SDK
  - Xpand SDK

TFD Projekt auf die Codegenerierung vorbereiten

- **Herunterladen von CodeGenClasses.ctr aus dem Blog**
  - Enthält notwendige Klassen aus dem MWE Framework als Referenz
- **Hinzufügen der CodeGenClasses.ctr zum TFD Projekt**
- **CodeGenClasses.ctr als Abhängigkeit in eurer tfdModel.ctr angeben**
  - Genau wie bei **`JavaClasses`** und **`EclipseClasses`**
- **In die Manifest.MF zu den Plugin Dependencies hinzufügen:**
  - **`org.eclipse.emf.mwe.core`**

Action für die Codegenerierung anlegen

- **Eine neue Klasse für die Codegenerierung anlegen**
  - Erbt von **`TransactionActionDelegate`**
  - **`runImpl`** implementieren
- **Als Selektion soll die Action ein Diagrammobjekt akzeptieren**

Implementierungsdetails runImpl - I

1. Diagram aus der `IStructuredSelection` holen (wie gehabt)
2. StatementActivity für das Setzen notwendiger MWE Parameter:

```
//define parameters for MWE
String resourceUri = diagram.eResource().getURI().toPlatformString(false);
String pathUri = resourceUri.subSequence(0,
                 resourceUri.lastIndexOf('/')).toString() + "/generated";

String fileString =
ResourcesPlugin.getWorkspace().getRoot().getLocation().toString() + pathUri;

Map<String,String> properties = new HashMap<String,String>();
Map slotContents = new HashMap();
slotContents.put("model", diagram);
properties.put("srcGenPath", fileString);

final String WORKFLOW_FILE = "generatorWorkflow.mwe";
```

Implementierungsdetails runImpl - II

3. Neues `NullProgressMonitor` Objekt anlegen

- Klasse kommt aus `CodeGenClasses.ctr`

4. Neues `WorkflowRunner` Objekt anlegen

- Klasse kommt aus `CodeGenClasses.ctr`

5. Auf dem `WorkflowRunner` mittels CollaborationStatement

```
run(WORKFLOW_FILE, monitor, properties, slotContents)
```

aufrufen

Optional Folders refreshen

6. Project refreshen

```
// refresh folder
IResource project =
        ResourcesPlugin.getWorkspace().getRoot().findMember(pathUri);
if (project != null)
{
    try {
        project.refreshLocal(IResource.DEPTH_INFINITE, null);
    } catch (CoreException e) {
        e.printStackTrace();
    }
}
```

Benötigte Klassen importieren

- **Klassen aus den Statement Activities müssen per Hand in die `GenerateCodeAction` importiert werden**
  - Klasse im Klassendiagramm selektieren
  - Rechtsklick -> Edit Imports
    - `java.util.Map`
    - `java.util.HashMap`
    - `org.eclipse.core.resources.ResourcesPlugin`
    - `org.eclipse.core.resources.IResource`
    - `org.eclipse.core.runtime.CoreException`
  - Falls auf linker Seite nicht bereits vorhanden über new -> Auswahl Class hinzufügen und auf die rechte Seite des Dialogs übernehmen.
  - Projekt speichern.

Workflow file anlegen

- **In den src Ordner eures Projektes eine Workflow Datei anlegen:**
  - Rechtsklick -> New -> Other -> Modeling Workflow Engine -> Workflow File
  - Gleichen Namen verwenden wie in der Statement Activity eurer `GenerateCodeAction`, sonst wird die Datei nicht gefunden.

Inhalt der Workflow Datei

```
<?xml version="1.0"?>
<workflow>
    <component id="generator" class="org.eclipse.xpand2.Generator" skipOnErrors="true">
        <fileEncoding value="ISO-8859-1"/>
        <metaModel id="mm" class="org.eclipse.xtend.typesystem.emf.EmfMetaModel">
            <metaModelPackage value="#vollqualifizierter Name eurer Package Klasse"/>
        </metaModel>

        <outlet path="${srcGenPath}">
            <postprocessor class="org.eclipse.xpand2.output.JavaBeautifier"/>
        </outlet>

        <!--protected regions configuration -->
        <prSrcPaths value="${srcGenPath}"/>
        <prDefaultExcludes value="false"/>

        <expand value="TFDDefault::main FOR model"/>
    </component>
</workflow>
```

Wert wird in Statement Activity gesetzt

„model“ wird in Statement Activity gesetzt

Name der Template Datei ist TFDDefault,

Name des Templates darin ist main

Templatedatei anlegen

- **Im src Ordner eures Projektes: New -> Other -> Xpand -> Xpand Template**
  - Name muss dem im Workflow file angegebenen entsprechen
- **Importieren eures Metamodells ins Template**

```
«IMPORT model»
```

- **Zugriff auf die Elemente des Metamodells über Code Completion möglich**

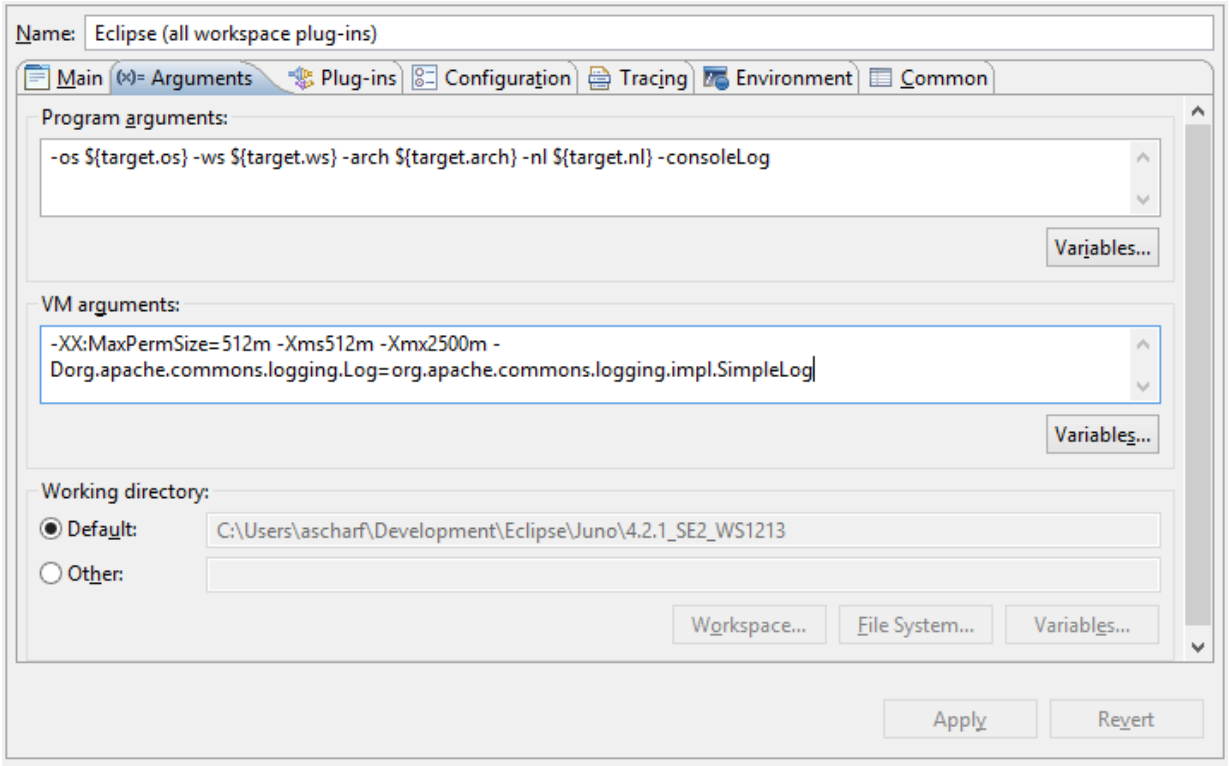
```
«DEFINE main FOR model::Diagram»

```

- **Genaue Syntax und Verwendung der Xpand language unter:**

  http://www.openarchitectureware.org/pub/documentation/4.3.1/html/contents/core_reference.html#xpand_reference_introduction

Troubleshooting

- **Für mehr Debug Output während dem Generieren:**
  - Run Konfiguration anpassen:
    - -Dorg.apache.commons.logging.Log=org.apache.commons.logging.impl.SimpleLog



Hausaufgabe 7

- **Formale Aufgabenstellung folgt bis heute Abend**
- **Abgabe bis zum 16.02.2013, 23:59 Uhr**