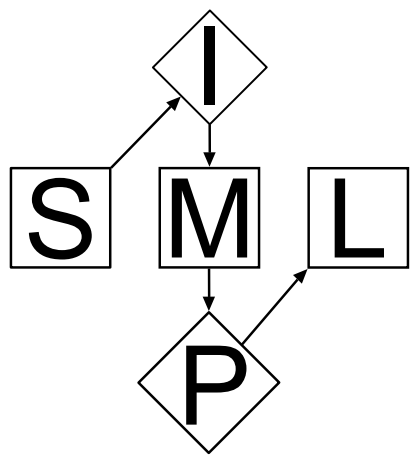
Wochenbericht

Firas Zoabi

Studienprojekt A: SIMPL

28. Dezember 2009



Geplante Aufgaben und Tätigkeiten

1. Untersuchung der Implementierung von M.Hanh bezüglich der Extension-Points.
2. Konzept entwerfen ,wie es möglich wäre die Abfragesprache-Befehlswörter (z.b Insert,where ...) mit eine bestimmte Logik bzw. Regeln festzulegen wie :
- welches Abfragesprache-wörter am zuerst kommen müssen
- welche und wie viele Abfragesprache-Wörter bzw. schlussel wörter dürfen nach welche auftauchen
- welche Restrektion und Regeln mussen erfühlt sein
3. Der Konzept baut sich anhand der Extension-Point auf ,sodass man einen XML Datei in der Extension definiert bzw. refereziert. Und dieses XML datei beinhaltet die gewünschten Schlüßel-Wörter mit der gewünschten definierten Logik .
Erste Realisierung dieses Konzept ist fertig. so habe ich das Implmentiert und es funktioniert :) .

Erledigte Aufgaben und Tätigkeiten

Übersicht

- Bearbeitung der PlugIn ExtensionPoint Entwicklung und überlegung "wie ist dies für die PropertieFenster entwickelt soll.
- Lösung von Tech.Problemen wegen importieren der Projekte vom SVN und Behebung der aufgetretene Fehlern seitens Eclipse.
- Code durchgehen ,und die Klassen von Hahn ersmal verstehen.
- Implementierung der ExtensionPoint mit einen neuen Attribute "queryStatmentsLogikXmlFile" der den Logik der Abfrage Befehle definiert und durch Extensions die GUI Elemente der Editor verändert bzw. bestimmt.
- Es wurde der Kern funktionalität erreicht bzw. entwickelt ,sodass die SQL Befehle aus dem xmlFile der Extension geparrst und als Objekte zwischen gespeichert.
- weiter Entwicklung bzw. Implementierung der Konzept.
-Erste Realisierung dieses Konzept ist fertig. so habe ich das Implmentiert und es funktioniert :) .

Erledigte Aufgaben und Tätigkeiten

Investierte Arbeitszeit/Aufwände

- 21.Dez : 5
- 24.Dez : 5,5
- 26.Dez : 5,75
- 27.Dez : 6,5

Gewonnene Erkenntnisse

- Es ist möglich mit voller Usabillity für den User, den DM-PropertieGUI über den Extension-Point und damit der referensierte xmlDatei zu erweitern .

Neues für SIMPL

Diagramme, Screenshots, usw.

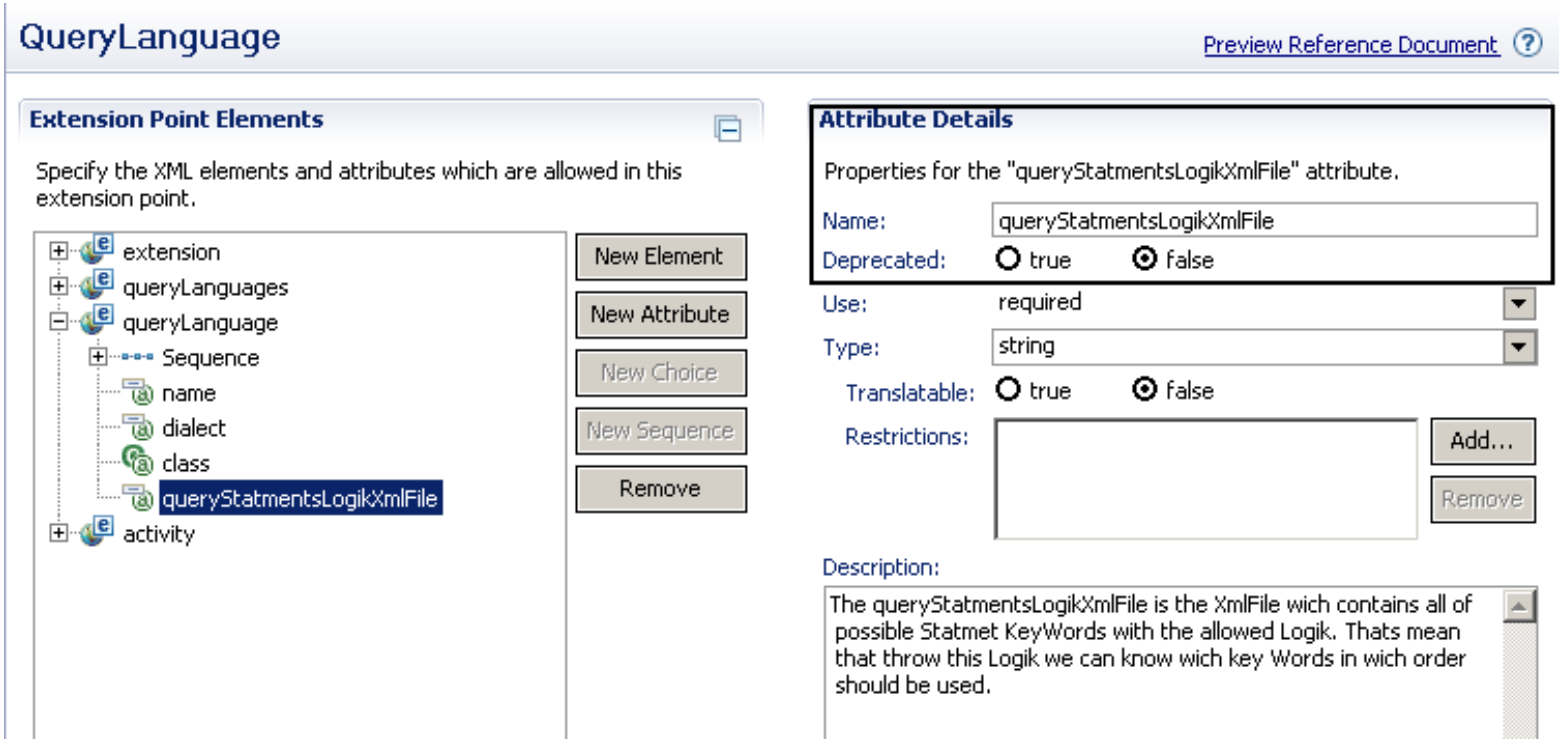


Abbildung: 1 Extension-Point QuaryLanguage

```
      <attribute name="class" type="string" use="required">
         <annotation>
            <documentation>
               The class which must extend org.eclipse.bpel.simpl.ui.extensions.AState
            </documentation>
            <appinfo>
               <meta.attribute kind="java" basedOn="org.eclipse.bpel.simpl.ui.extensio
            </appinfo>
         </annotation>
      </attribute>
      <attribute name="queryStatmentsLogikXmlFile" type="string" use="required">
         <annotation>
            <documentation>
               The queryStatmentsLogikXmlFile is the XmlFile wich contains all of poss
            </documentation>
         </annotation>
      </attribute>
   </complexType>
</element>
```

Abbildung: 2 ExtensionPoint XMl-Struktur mit dem neuen Attribute für XMlDatei

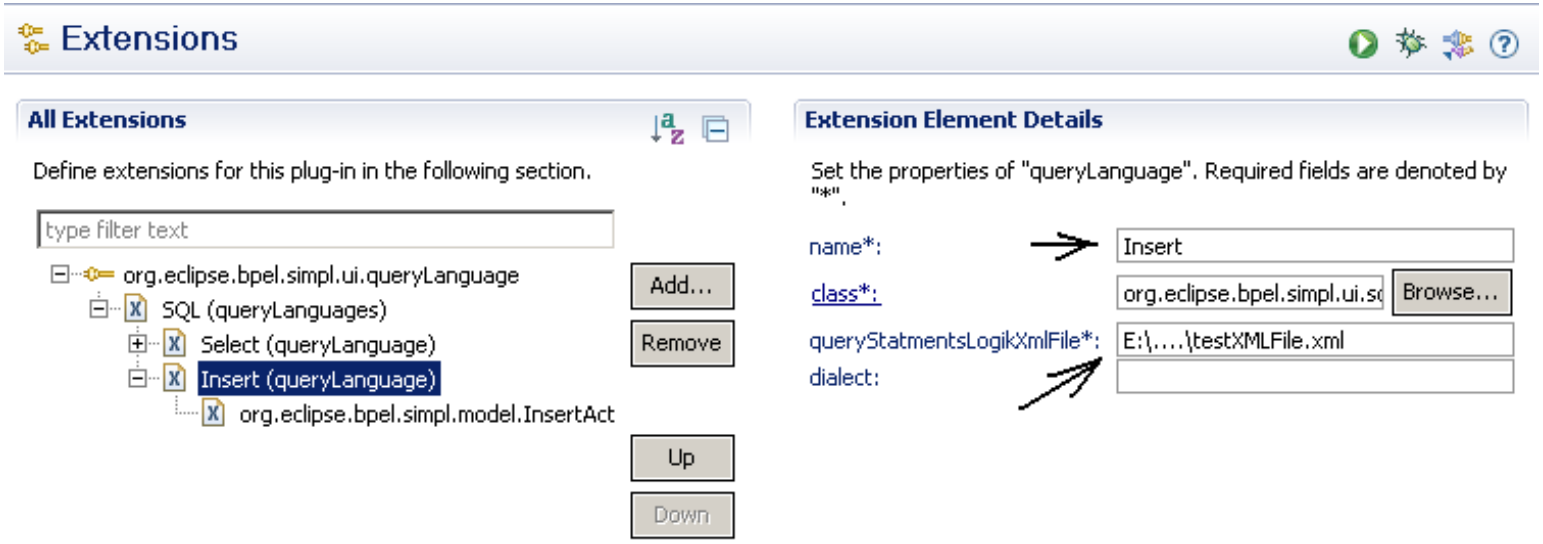


Abbildung: 3 Das Extension User Interface. achte auf der pfad und name des Extensions

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.4"?>
<plugin>

    <extension
          point="org.eclipse.bpel.simpl.ui.queryLanguage">
       <queryLanguages
             language="SQL">
          <queryLanguage
                class="org.eclipse.bpel.simpl.ui.sql.editor.SelectEditor"
                name="Select"
                queryStatmentsLogikXmlFile="dddddd">
             <activity
                   type="org.eclipse.bpel.simpl.model.QueryActivity">
             </activity>
          </queryLanguage>
          <queryLanguage
                class="org.eclipse.bpel.simpl.ui.sql.editor.InsertEditor"
                name="Insert"
                queryStatmentsLogikXmlFile="E:\....\testXMLFile.xml">
             <activity
                   type="org.eclipse.bpel.simpl.model.InsertActivity">
             </activity>
          </queryLanguage>
       </queryLanguages>
    </extension>
```

Abbildung: 4 Die Komplette Extensions bzw. PlugIn

In Abbildung 4 sehen wir die Extensions xml baum. Gehen wir mal die Datei von oben nach unten durch.
-Element <quaryLanguages> definiert eine sprache wie SQLx etc. und der name der Abfragesprache kommt in the Attribute “language” (siehe im abbildung 4 language=”SQL”).
-im <quaryLanguage> definieren wir der DM-Aktivität der diesen Abfragesprache verwenden soll und dazu werden die drei Attribute mit die Klasse, Name und der referenzierte XML-Datei, der alle Schlüßel-Wörter beinhaltet sowie der Logik, definiert. Die drei Attribute sind:

- class= “....”
- name= “...”
- quaryStatmentsLogikXmlFile= “c:\ ....”

(siehe Abbildung 5.)

```
            <activity
                  type="org.eclipse.bpel.simpl.model.QueryActivity">
            </activity>
         </queryLanguage>
         <queryLanguage
               class="org.eclipse.bpel.simpl.ui.sql.editor.InsertEditor"
               name="Insert"
               queryStatmentsLogikXmlFile="E:\....\testXMLFile.xml">
            <activity
                  type="org.eclipse.bpel.simpl.model.InsertActivity">
            </activity>
         </queryLanguage>
      </queryLanguages>
   </extension>

</plugin>
```

Abbildung: 5 Die Extenstion über QuaryLanguage ExtensionPoint

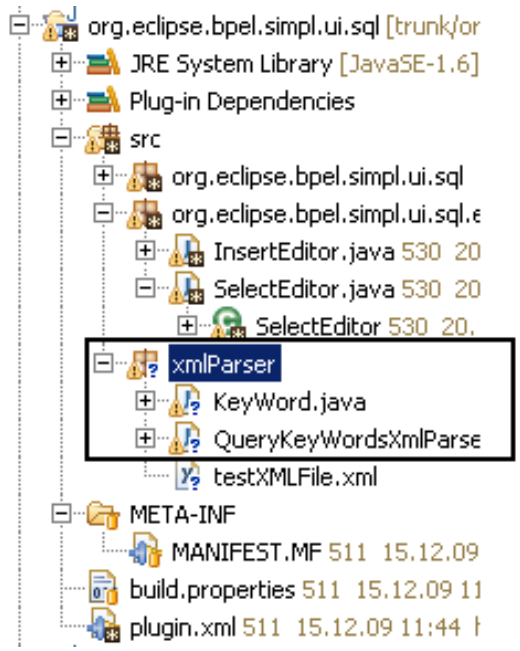


Abbildung: 6 XML Parser Klassen

Hier können wir den Package “xmlParser”, der die Klasse QueryKeyWordsXmlPraser.java beinhaltet. Diese Klasse parsst der XML datei (siehe Abbilsung 7: Der Xml Datei), welche die Abfragesprache-Logik sowie die Schlüßelwörter beinhaltet, parsst. Daraus werden objekte generiert, die zur Gestaltung der überfläche der StatmentEditor gebraucht werden.

testXMLFile.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<root>
 <MainKeyWord value="Insert">
    <KeyWord value="In">
        <KeyWord value="In x">
            <KeyWord value="In xx">
            </KeyWord>
        </KeyWord>
    </KeyWord>
    <KeyWord value="where">
        <KeyWord value="where x">
        </KeyWord>
        <KeyWord value="where y">
        </KeyWord>
        <KeyWord value="where z">
        </KeyWord>
    </KeyWord>
    <KeyWord value="Z">
        <KeyWord value="Zz">
        </KeyWord>
        <KeyWord value="Zz2">
        </KeyWord>
    </KeyWord>
 </MainKeyWord>
</root>
```

Abbildung: 7 Die Xml Datei

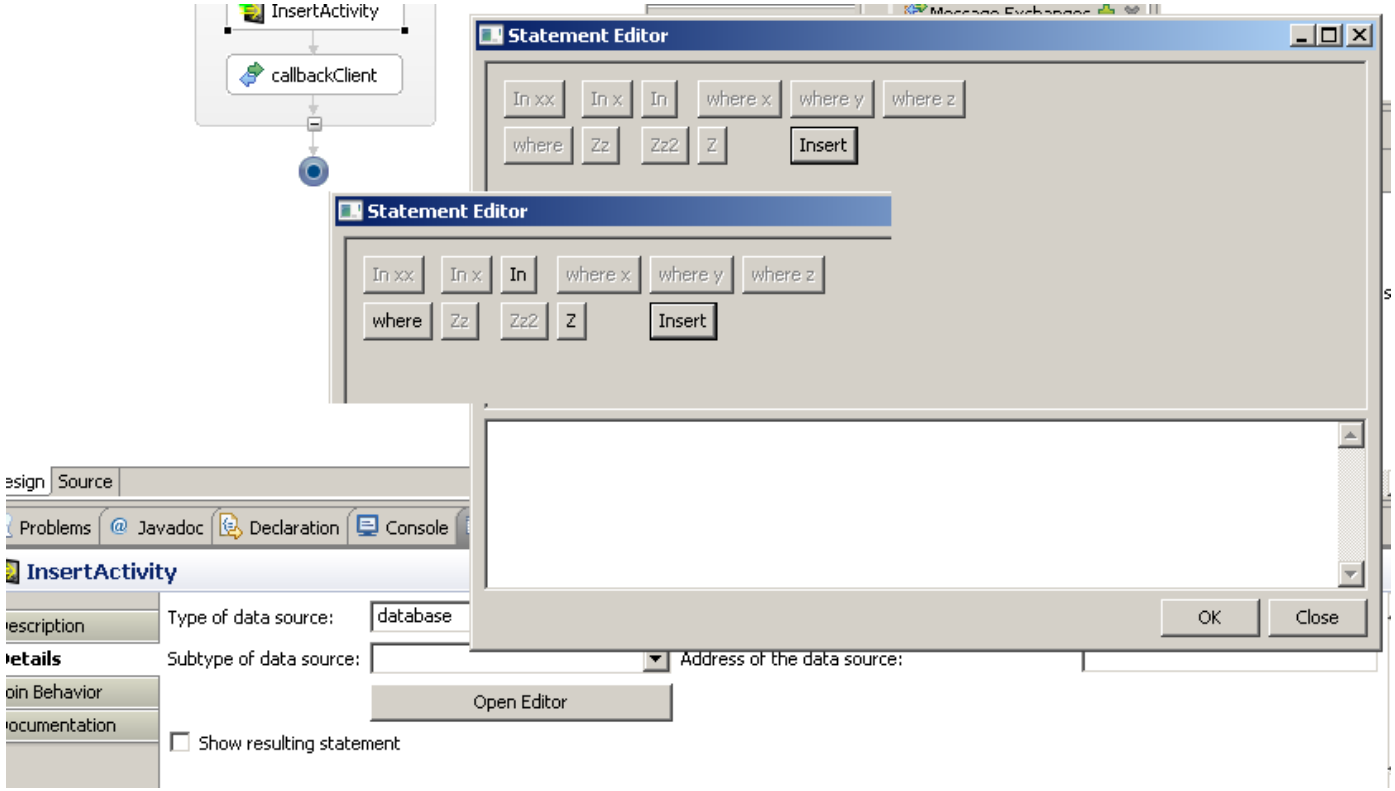


Abbildung: 8 Ergebnis des Extension

Im Abbildung 8 wird die Ergebniss der Extension im Editor dargestellt. Diese Buttons mit die Schlüßelwörter werden dann anhand der xmlDatei, der vom Extension referenziert wurde.
Wie wir auch es sehen, dass die Schlüßelwörter können nur in einer defenierte (wie im xml definiert ist) angecklickt. Das heißt wenn erst Insert gecklickt wird ,dann werden die Befehle zweite stufe erst freigeschaltet. etc. so wird die Logik der Statment gewährleistet.
Man kann natürlich weitere regeln im Xml Datei definieren und vom GUI bearbeitet werden.

Verwendete Quellen/Literatur

keins

Identifizierte Aufgaben für die kommende Woche

- Weitere Impelementierung der ExtensionPoint von PropertieFensetr der Aktivitäten.
- Kommentieren des code von meiner und von M.Hahns Klassen.
- Alle unnötigen Code wegentfernen
- Kompfkommentare hinzufügen.
- Weiter Implementierung der Load und save methoden.

- @M.Hahn: was kommt eigentlich alles in der Editor ?
- Muss auch die Tabellen und Spalten einer DB zum Beispiel live abgerufen und in der Propertiesfenster der DM-PlugIn angezeigt werden ?