Transformations

Die API des Oracle Datamodeler

Dr. Gudrun Pabst

Trivadis GmbH
Lehrer-Wirth-Straße 4
81829 München

gudrun.pabst@trivadis.com

BASEL BERN LAUSANNE ZÜRICH DÜSSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MÜNCHEN STUTTGART WIEN

trivadis
makes IT easier.

AGENDA

- **Transformationen**
- Erstellen eigener Skripte
  - Elemente eines Skripts
  - Ausgabe von Text
  - Interaktion mit dem Aufrufer
  - Programmieren des Skripts
- Libraries
- Zusammenfassung

Einführung

- Oracle Data Modeler
  - hier als Extension im SQL Developer
  - Routinearbeiten bei der Datenmodellierung
    - Anlegen von Audit-Spalten
    - Erstellen von Triggern zum Füllen der Audit-Spalten
    - Sicherstellen von Namenskonventionen
    - …
  - Automatisierung der Routinearbeiten
    - Arbeitserleichterung für den Modellierer
    - höhere Qualität des Datenmodells
  - „Transformations“
    - Selbstprogrammierte Skripte zum Durchführen der Routinearbeiten
    - Verschiedene Skript-Sprachen verwendbar

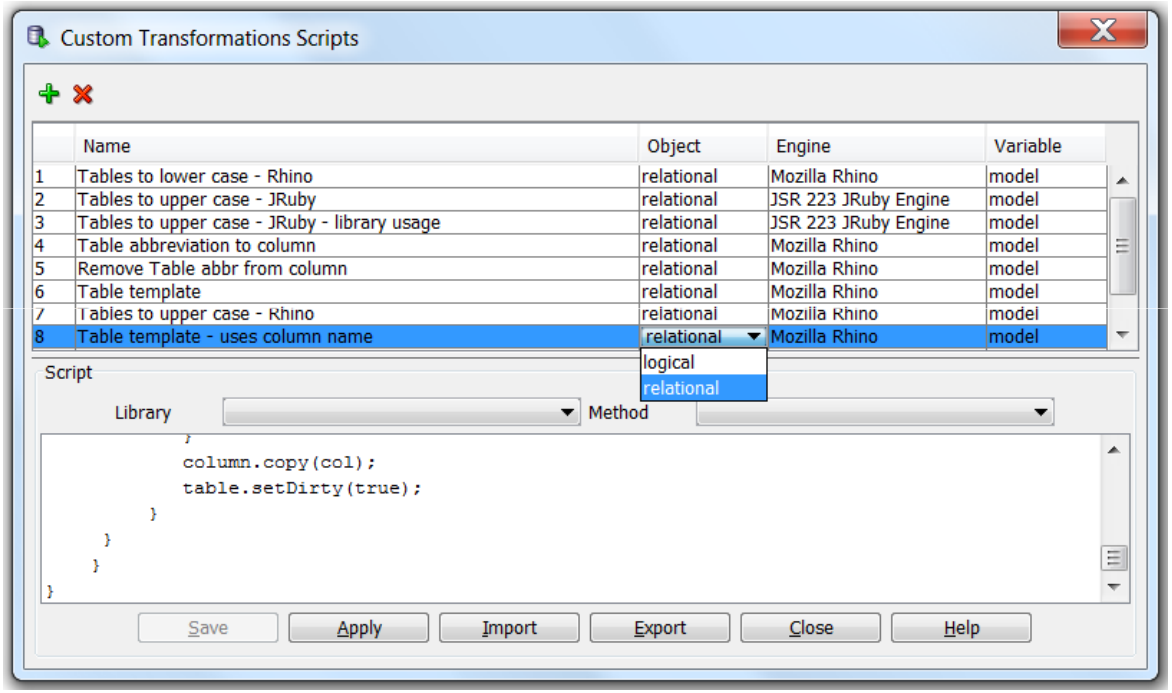
Grundlagen

- „Scripting for Java“ (JSR 233)
  - Einbinden von Skript-Sprachen in Java-Programme
  - Erweiterung der Programme durch eigene Funktionalitäten möglich
  - Beispiele für Implementierungen von Skript-Sprachen:
    - Mozilla Rhino
      - Anbindung von JavaScript
      - in JDK 6 und höher bereits enthalten
      - Scripting Engine steht daher im Data Modeler bereits zur Verfügung
    - JRuby
      - Anbindung von Ruby
      - Beispielskript wird mitgeliefert
      - Scripting Engine muss erst in den Data Modeler eingebunden werden
    - Caucho Quercus Script Engine
      - Anbindung von PHP
      - Scripting Engine muss erst in den Data Modeler eingebunden werden

Transformations

- Data Modeler als Extension des Oracle SQL Developer
  - Anzeige der Transformations:

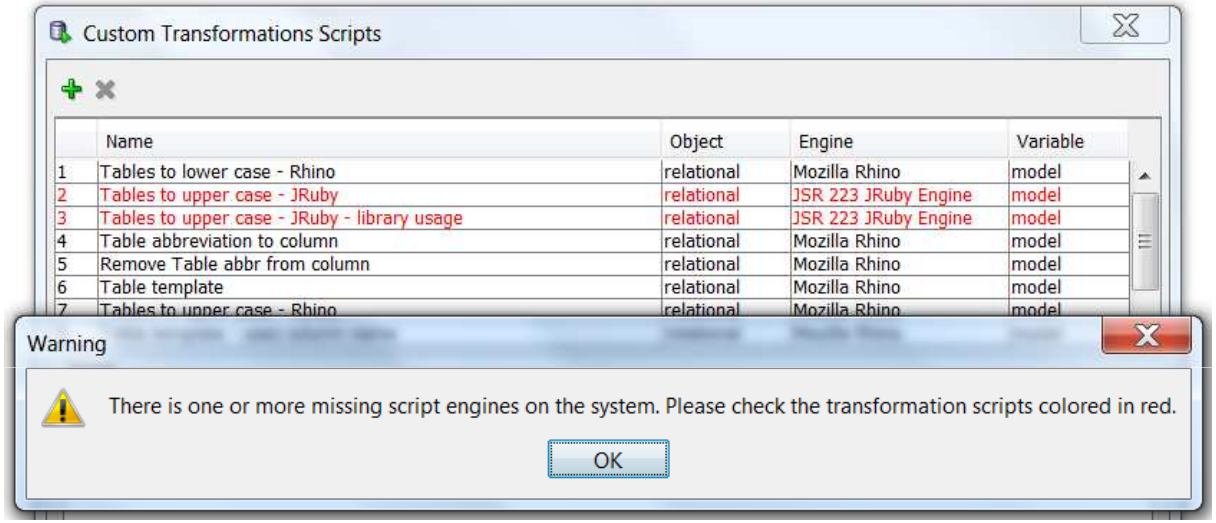
    Tools → Data Modeler → Design Rules → Transformations



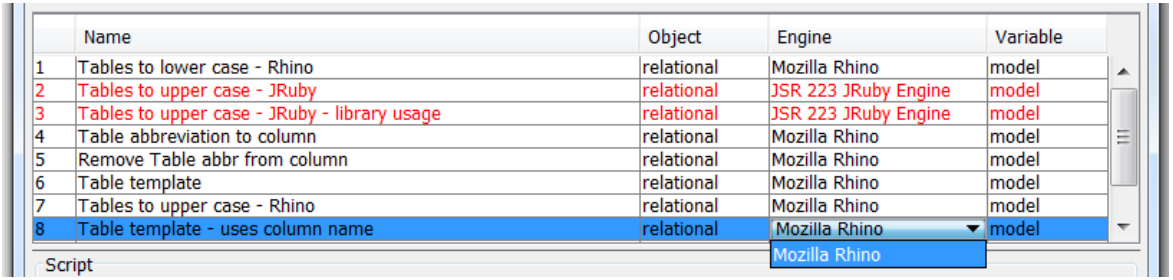
  - „Object“ : logische oder relationale Ebene (nur Dokumentation)
  - „Engine“ : Auswahl der Scripting Engine zum Ausführen des Skripts

Einbinden von Scripting Engines

- Aufruf der Transformations ohne weitere Scripting Engines
  - Warnhinweis, da die JRuby Engine nicht eingebunden ist

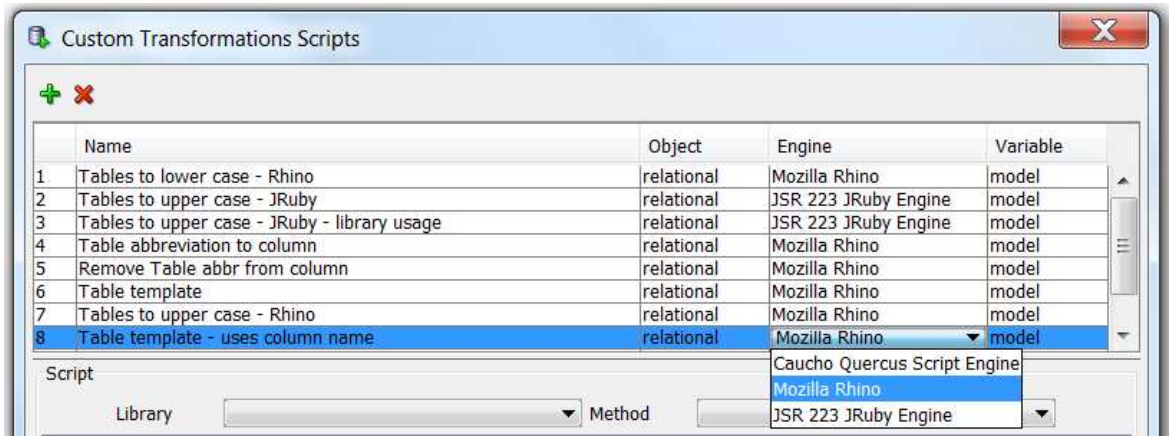


  - Unter „Engine“ nur „Mozilla Rhino“:



Einbinden von Scripting Engines

- Vorgehen
  - Voraussetzung: JAR-Datei der Scripting Engine ist vorhanden
    - JRuby : `jruby.jar`
    - Quercus : `resin.jar`
  - Ermitteln der vom SQL Developer verwendeten JVM
  - Kopieren der JAR-Datei in das Verzeichnis `jre\lib\ext` dieser JVM
  - Neustart des SQL Developer



Mitgelieferte Dokumentation

- Verzeichnis

  …\sqldeveloper\sqldeveloper\extensions
  \oracle.datamodeler\xmlmetadata\doc

  - README.rtf Kurzbeschreibung zum Umgang mit der API
  - index.html Master für die Navigations- und die Inhaltsseite; Beschreibung des XML-Formats des Datenmodells

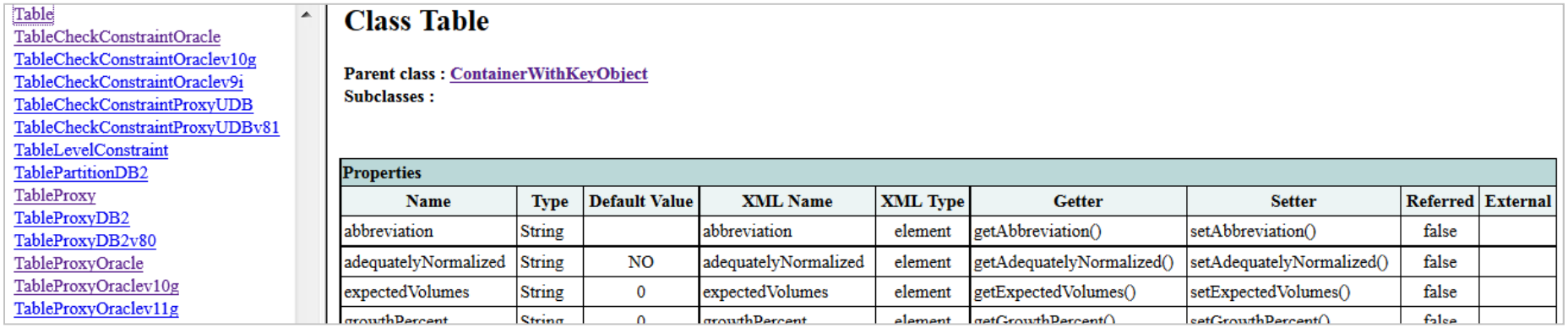


Table
TableCheckConstraintOracle
TableCheckConstraintOraclev10g
TableCheckConstraintOraclev9i
TableCheckConstraintProxyUDB
TableCheckConstraintProxyUDBv81
TableLevelConstraint
TablePartitionDB2
TableProxy
TableProxyDB2
TableProxyDB2v80
TableProxyOracle
TableProxyOraclev10g
TableProxyOraclev11g

Class Table

Parent class : ContainerWithKeyObject
Subclasses :

Properties								
Name	Type	Default Value	XML Name	XML Type	Getter	Setter	Referred	External
abbreviation	String		abbreviation	element	getAbbreviation()	setAbbreviation()	false	
adequatelyNormalized	String	NO	adequatelyNormalized	element	getAdequatelyNormalized()	setAdequatelyNormalized()	false	
expectedVolumes	String	0	expectedVolumes	element	getExpectedVolumes()	setExpectedVolumes()	false	
growthPercent	String	0	growthPercent	element	getGrowthPercent()	setGrowthPercent()	false	

AGENDA

- Transformationen
- **Erstellen eigener Skripte**
  - Elemente eines Skripts
  - Ausgabe von Text
  - Interaktion mit dem Aufrufer
  - Programmieren des Skripts
- Libraries
- Zusammenfassung

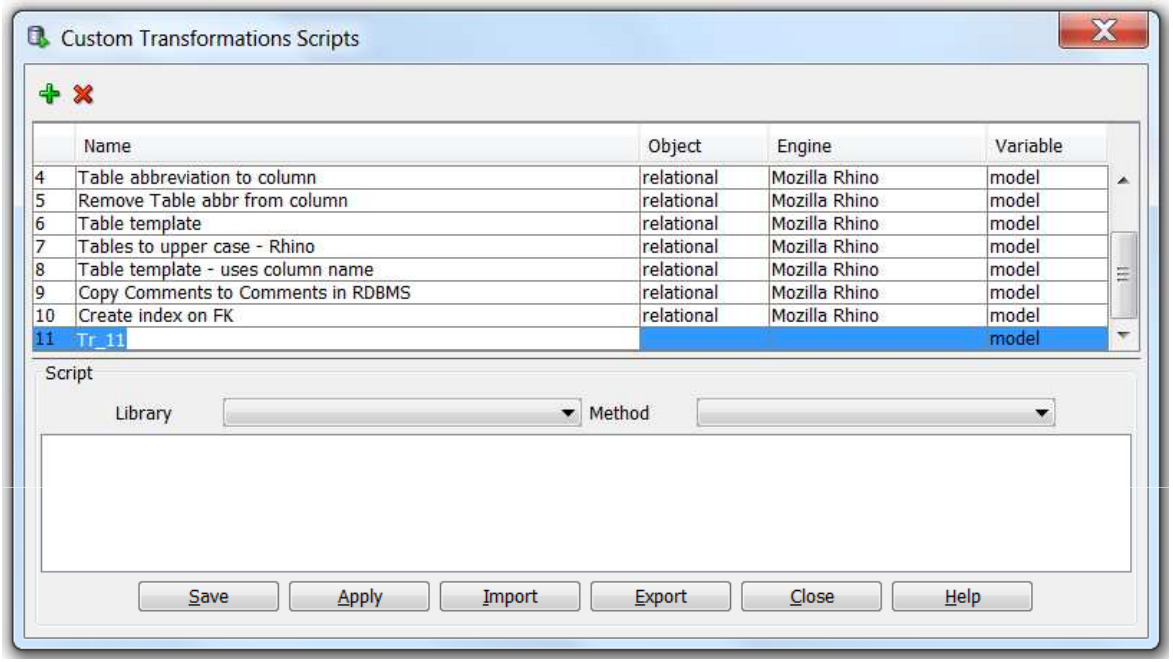
Mitgeliefertes Beispiel

- „Table template - uses column name“
  - Anfügen von Spalten einer Template-Tabelle an die anderen Tabellen

```
var t_name = "table_template";
template = model.getTableSet().getByName(t_name);
if(template!=null){
    tcolumns = template.getElements();
    tables = model.getTableSet().toArray();
    for (var t = 0; t<tables.length; t++){
     table = tables[t];
     // compare name ignoring the case
     if(!table.getName().equalsIgnoreCase(t_name)){
         for (var i = 0; i < tcolumns.length; i++) {
            column = tcolumns[i];
            col = table.getElementByName(column.getName());
            if(col==null){
             col = table.createColumn();
            }
            column.copy(col);
            table.setDirty(true);
        } } } }
```

Einbinden eigener Skripte

- Anlegen eines neuen Skripts:



  - Angabe eines sprechenden Namens und der gewünschten Engine
  - Eingabe des Skripts im Code-Bereich
  - Speichern mit „Save“
  - Ausführen mit „Apply“

Ausgabe von Text zur Kontrolle der Verarbeitung

- Benachrichtigung des Bearbeiters über Meldungsfenster
  - Mozilla Rhino: serverseitiges JavaScript, daher kein `alert`
  - „Scripting for Java“: Java-Umgebung steht im Skript zur Verfügung
  - Mozilla Rhino: Einbinden von Java-Paketen über `importPackage`
    Einbinden von Java-Klassen über `importClass`
  - Ausgabe von Nachrichten über javax.swing.JOptionPane:

```
importClass(javax.swing.JOptionPane);
var vText = "Demo der Ausgabe von Text";
JOptionPane.showMessageDialog(null, vText);
```

- Nur für kurze Texte geeignet

Ausgabe von Text zur Kontrolle der Verarbeitung

- Logging von Informationen in einer Note
  - Anlegen einer Note im Modell
  - Leider kein Zugriff auf die Eigenschaften einer Note über die GUI
  - Daher: Finden der Note über ein Skript
    Angabe eines sprechenden Namens: `setName("Ausgabe")`
  - Ausgabe der Informationen:

```
var vText = "";
var vNote = model.getNoteSet().getByName("Ausgabe");
...
vNote.setComment(vText);
```

Interaktion mit dem Bearbeiter

- Verschiedene Methoden von `javax.swing.JOptionPane`

- Wahlmöglichkeit für den Bearbeiter: `showConfirmDialog`

```
importClass(javax.swing.JOptionPane);
var vAntwort = JOptionPane.showConfirmDialog (
                  null, "Weitermachen?", "Weitermachen?",
                  JOptionPane.YES_NO_OPTION );
```

- Eingabemöglichkeit für den Bearbeiter: `showInputDialog`

```
importClass(javax.swing.JOptionPane);
var vTable = JOptionPane.showInputDialog (
                  null, "Eingabe der Tabelle", "Tabelle",
                  JOptionPane.PLAIN_MESSAGE );
```

Programmieren des Skripts

- Problem: Anlegen eines Check Constraint für jede Tabelle mit Spalten `<xxx>_GUELTIG_VON` und `<xxx>_GUELTIG_BIS`
- Ablauf des Skripts:
  - Durchlaufen aller Tabellen des Modells: **`model.getTableSet()`**
  - Durchlaufen aller Spalten der aktuellen Tabelle: **`Tab.getElements()`**
    Namensvergleich: **`Col.getName()`**
    ggf. Speichern des Namens in der zugehörigen Variable
  - Nach Durchlaufen der Spalten Prüfung, ob der Check Constraint benötigt wird
  - Prüfung, ob Check Constraint vorhanden:
    `Tab.getCheckConstraints()`
  - Ggf. Anlegen des Check Constraint:
    `Tab.createCheckConstraint()`
    `Tab.addCheckConstraint(…)`

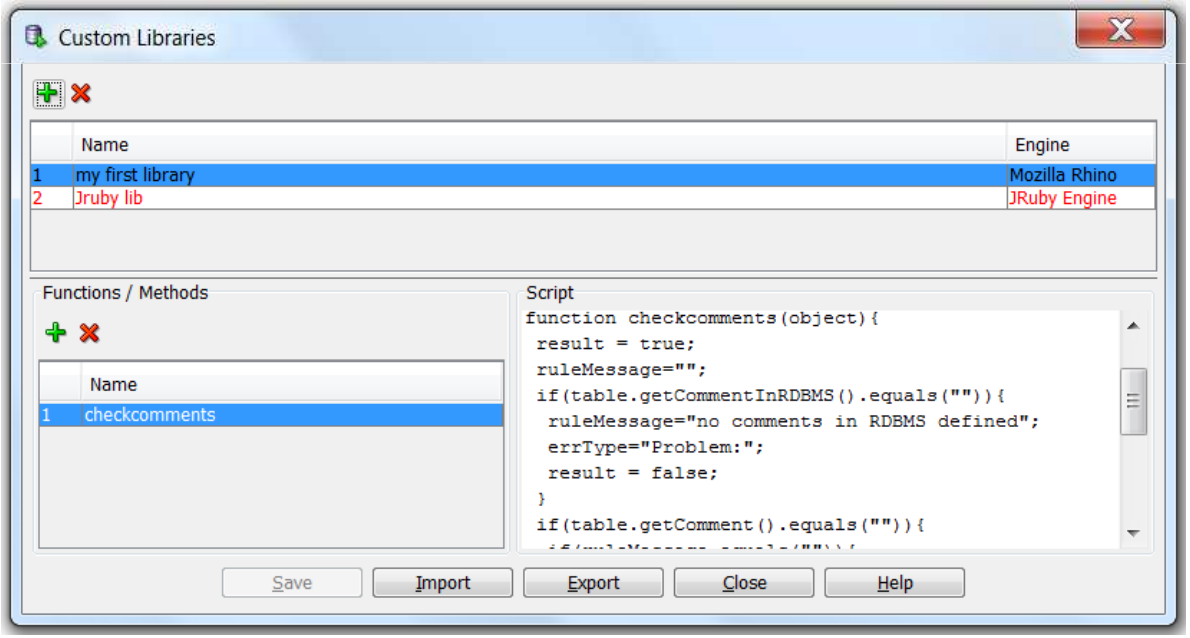
AGENDA

- Transformationen
- Erstellen eigener Skripte
  - Elemente eines Skripts
  - Ausgabe von Text
  - Interaktion mit dem Aufrufer
  - Programmieren des Skripts
- **Libraries**
- Zusammenfassung

Libraries

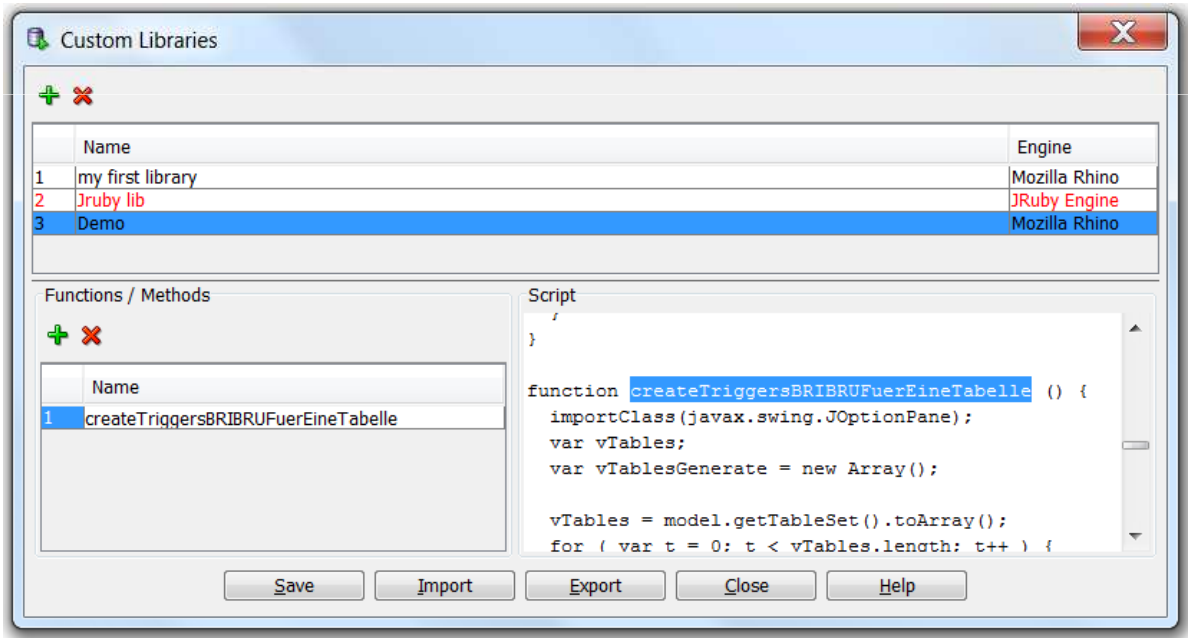
- Skripte können zu Libraries zusammengefasst werden:
  - Skript wird als JavaScript-Funktion in einer JavaScript-Datei abgelegt
  - Verwendung von vorher in der Datei deklariertem Code möglich
- Libraries:

  Tools → Data Modeler → Design Rules → Libraries



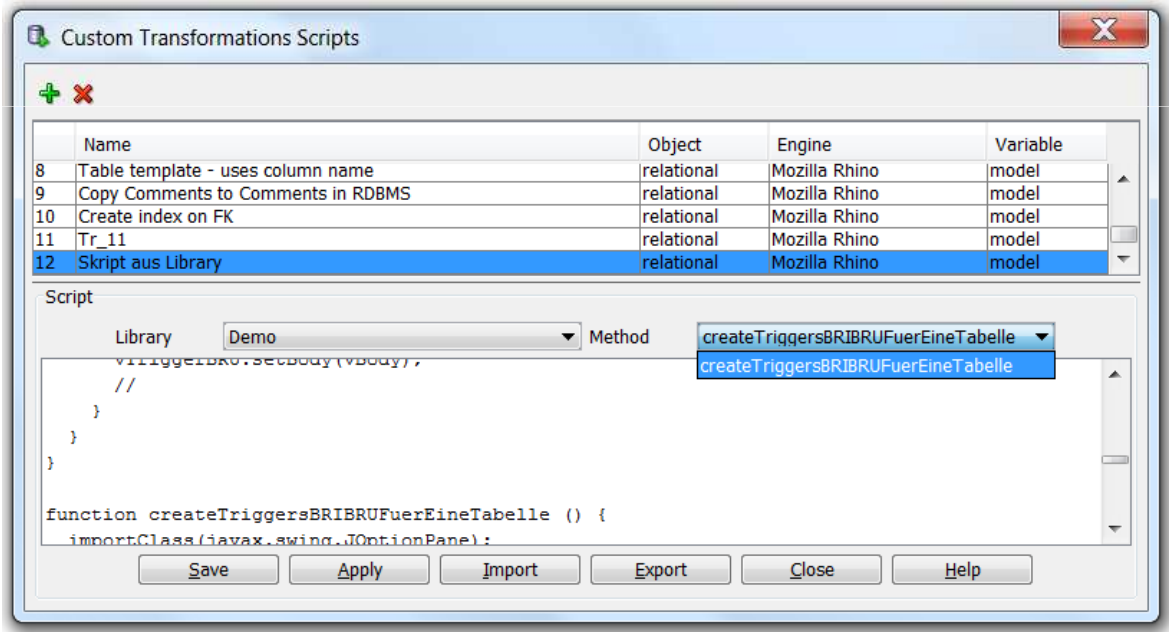
Libraries

- Einbinden von Libraries
  - Anlegen einer Zeile, Angabe eines Namens und der Scripting Engine
  - Eintragen des Library Code unter „Script“
  - „Functions / Methods“:
    Veröffentlichen der gewünschten Funktion durch Angabe des Namens



Skripte aus Libraries

- Transformations
  - Anlegen eines leeren Skripts
  - Auswahl der Library → Code der Library wird angezeigt
  - Auswahl der gewünschten Funktion → nur veröffentlichte Funktionen auswählbar



AGENDA

- Transformationen
- Erstellen eigener Skripte
  - Elemente eines Skripts
  - Ausgabe von Text
  - Interaktion mit dem Aufrufer
  - Programmieren des Skripts
- Libraries
- **Zusammenfassung**

Zusammenfassung

- Transformations
  - Einfache Erweiterungsmöglichkeit des Oracle Data Modeler um Skripte
  - Einfaches Einbinden weiterer Scripting Engines → bevorzugte Skript-Sprache kann benutzt werden
  - Automatisierung von Standardaufgaben
  - Organisation von Skripten in Libraries möglich
  - Libraries erlauben Wiederverwendung von Code

VIELEN DANK

Dr. Gudrun Pabst

Trivadis GmbH
Lehrer-Wirth-Straße 4
81829 München

gudrun.pabst@trivadis.com

BASEL BERN LAUSANNE ZÜRICH DÜSSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MÜNCHEN STUTTGART WIEN