Beispiele für Regel-Implementationen von ProxyBOs

ProxyBOs können nur in Unterformularen verwendet werden. Dafür muss also wie bei anderen BOs auch ein Referenzfeld auf das BO in dem das ProxyBO als Unterformular angezeigt werden soll, erstellt werden. Es ist auch nicht möglich ein Referenzfeld auf ein ProxyBO bereitzustellen.

Erster Fall: Die Rechnungen der Kunden werden nicht in der Nuclos-Datenbank gespeichert, sondern werden auf einem Drittsystem gepflegt. Dazu kann ein sogenanntes "ProxyBO" Rechnungen im Bo-Wizard erstellt werden. Damit wird ein Interface (hier: "org.nuclet.businessentity.RechnungProxy", wichtig ist das komplette Ausschreiben des Pfades, damit die Klasse aus Proxy-Implementation erkannt wird) zur Verfügung gestellt, dessen Implementation die Daten holt, bzw. schreibt:

```
package org.nuclet.businessentity;
import java.util.*;
import org.nuclos.api.exception.*;
import org.nuclos.api.rule.*;

public class RechnungProxyImpl implements org.nuclet.businessentity.RechnungProxy {
private User user;

public List<Rechnung> getByKunde(Long id) {
  List<Rechnung> rlist = new ArrayList<>();
  // Get Rechnungen for Kunde with "id" from another source
  // ...
  // rlist.add(...)
  //
  return rlist;
}

public List<Rechnung> getAll() {
  // Get Data from another source
}
public List<Long> getAllIds() {
  // Get Data from another source
}
public Rechnung getById(Long id) {
  // Get Data from another source
}

public void insert(Rechnung rechnung) throws BusinessException {
  // Write Data to another source
}
public void update(Rechnung rechnung) throws BusinessException {
  // Write Data to another source
}
public void delete(Long id) throws BusinessException {
  // Delete Data within another source
}

public void commit() {
}
public void rollback() {
}
public void setUser(User user) {
  this.user = user;
}
}
```

Zweiter Fall (ab Nuclos 4.16): Es soll eine Sammelbearbeitung geben, bei der die Rechnungen mehrerer Kunden auf einmal bearbeitet werden. Grundsätzlich ist das schon mit dem Standard-Proxy-Interface oben möglich. Um die Performance zu verbessern, gibt es die Option eines weiteren Interface: "org.nuclos.api.rule.CollectiveProcessingProxy" fordert die Implementierung einer zusätzlichen Methode "getForCollectiveProcessing", mit der die Rechnungen mehrerer Kunden **auf einmal geholt** werden können:

```
package org.nuclet.businessentity;

import java.util.*;
import org.nuclos.api.businessobject.attribute.*;
import org.nuclos.api.exception.*;
import org.nuclos.api.rule.*;
import org.nuclos.api.*;

public class RechnungProxyImpl implements org.nuclet.businessentity.RechnungProxy, CollectiveProcessingProxy {
private User user;

public <PK> List getForCollectiveProcessing(ForeignKeyAttribute<PK> attribute, Collection<PK> ids) {
  if (attribute == Rechnung.KundeId) {
    List ret = new ArrayList<>();
    // Fetch Rechnungen für several Kunden (ids) at once
    // ret.add(...);
    //
    return ret;
  }
  return null; // When null is returned, the standard proxy methods (like getByKunde()) will be called
}

public List<Rechnung> getByKunde(Long id) {
  // Rest is the same as before
  // ...
```

Bitte beachten, dass mit dem Interface "CollectiveProcessingProxy" beim Laden von abhängigen Daten immer zuerst "getForCollectiveProcessing" aufgerufen wird. Nur wenn diese Methode "null" zurück gibt, wird die Standard-Methode "getByKunde" aufgerufen (bei mehreren ids in einer Schleife). Die vollständige Bezeichnung der Methode mit Generics lautet:

```
<PK> List<? extends BusinessObject<PK>> getForCollectiveProcessing(ForeignKeyAttribute<PK> attribute,
Collection<PK> ids);
```

Beispiel für Regel-Implementationen von Schreib-ProxyBOs

Schreib-ProxyBOs verhalten sich bzgl. der Lesens wie "normale" virtuelle BOs und können daher auch außerhalb von Unterformularen verwendet und referenziert werden.

Schreib-ProxyBOs sind eine abgewandelte Form der virtuellen BOs. Die Daten eines Schreib-ProxyBOs werden daher aus der dem BO hinterlegten Datenbank-View gelesen. Die Daten werden über die im Interface zu implementierenden Methoden geschrieben. Ein typischer Anwendungsfall für ein Schreib-ProxyBO ist ein externes System, in dem man neue Datensätze über Webservices anlegt, die aber eine Datenbank-View zum Lesen der Datensätze bereitstellt.

```
package example.rest;

public class ArtikelProxyImpl implements example.rest.ArtikelProxy{
        private User user;
        public Object insert(Artikel artikel) throws org.nuclos.api.exception.BusinessException{
            // Call Webservice
            WS myWS =...
            WSArtikel artikel=myWS.createArtikel(...);

            return artikel.getId();
      }
        public void update(Artikel  artikel) throws org.nuclos.api.exception.BusinessException{
               // Call Webservice
           WS myWS =...
           myWS.updateArtikel(...);
    }

        public void commit() {
        }
        public void rollback() {
        }
        public void setUser(User user) {
                  this.user = user;
        }
        public void delete(java.lang.Long id) throws org.nuclos.api.exception.BusinessException{}
}
```