Proxy Businessobjekt

Sprache: **Deutsch** · English

Proxy Businessobjekt

> **Externe Daten wie ein BO anzeigen und schreiben – Proxy-Interface, Sammelbearbeitung und Schreib-Proxy-BO.**

KONZEPT | ANWENDUNGSENTWICKLER | STAND: JUL 2026 | GILT FUER NUCLOS 4.2026.X

Auf dieser Seite

Was ist ein Proxy-BO?

Ein **Proxy Businessobjekt** wird verwendet, wenn Daten **nicht aus der Nuclos-Datenbank** stammen, sondern per **Java-Code aus externen Quellen** gelesen oder geschrieben werden – z.B. Drittsysteme, Webservices oder das Dateisystem.

Nuclos generiert dazu ein **Interface**, das in einer Regel implementiert wird. Diese Implementierung übernimmt Lesen, Schreiben und Löschen.

> ⚠ **Einsatz als Unterformular**
>
> Proxy-BOs können nur als **Unterformular** verwendet werden (Referenzfeld auf das Eltern-BO nötig). Ein Referenzfeld *auf* ein Proxy-BO ist nicht möglich.

How-to: Proxy-BO erstellen

1. **BO anlegen** (Konfiguration Businessobjekt) und Option *Proxy Businessobjekt* aktivieren – ein Java-Interface wird generiert.
2. **Attribute** definieren (ID, Name, Referenz auf das Eltern-Objekt …).
3. **Interface implementieren** (vollständiger Package-Pfad, z.B. `org.nuclet.businessentity.RechnungProxy`).
4. Implementierung **voll qualifiziert** registrieren.
5. Proxy-BO im Layout als Unterformular verwenden.

Beispiel: lesendes/schreibendes Proxy-BO

Die Methoden definieren, wie Nuclos mit den externen Daten interagiert (`getByKunde`, `getAll`, `getById`, `insert`, `update`, `delete`, `commit`/`rollback`):

```
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;
}
}
```

Sammelbearbeitung (CollectiveProcessingProxy, ab 4.16)

Für bessere Performance beim Laden abhängiger Daten implementiert das BO zusätzlich `CollectiveProcessingProxy`. `getForCollectiveProcessing` wird zuerst aufgerufen; nur bei Rückgabe `null` greift die Standardmethode (z.B. `getByKunde`) je Id.

```
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
  // ...
```

Schreib-Proxy-BO

Schreib-Proxy-BOs sind eine Variante der virtuellen BOs: **Lesen** erfolgt über die hinterlegte Datenbank-View (auch außerhalb von Unterformularen referenzierbar), **Schreiben** über die Interface-Methoden – z.B. via Webservice:

```
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{}
}
```

Verwandte Seiten

📦 Virtuelle Businessobjekte

Virtuelle BOs.

📖 View mit Dokumenten erstellen

Praxisbeispiel.

⚙️ Regeln (serverseitig)

Regeln.

Öffnen

Öffnen

Öffnen