View mit Dokumenten erstellen

Definition

Eine View mit Dokumenten erstellen.

Wenn man eine View erstellen möchte, in der Dokumente angezeigt werden, reicht es nicht aus, ein virtuelles oder generisches Businessobjekt zu erstellen.

Der Grund dafür ist, dass die Dateien nicht in der Datenbank abgelegt sind, sondern im Installationsverzeichnis des Nuclos-Servers (spezifisch **[nuclos-home]**/data/documents).

Sowohl virtuelle, als auch generische Businessobjekte können lediglich Daten aus der Datenbank anzeigen - hier sind aber nur die Namen und IDs der Dokumente hinterlegt.

Daher verwenden wir in diesem Fall ein *Proxy-BO* - üblicherweise wird dieses verwendet, wenn die Daten von außerhalb kommen.

Letzten Endes ermöglicht uns dieses Vorgehen, die View mit Java-Code zu generieren, weshalb wir die Dokumente aus dem Dateisystem laden können.

Bedenken Sie auch, dass Proxy-BOs nur als Subform an ein anderes Objekt (**[eltern-objekt]**) gehängt werden können - wenn es eigenständig sein soll, müssen Sie ein *Schreib-Proxy-BO* erstellen (Beispiele für Regel-Implementationen von ProxyBOs).

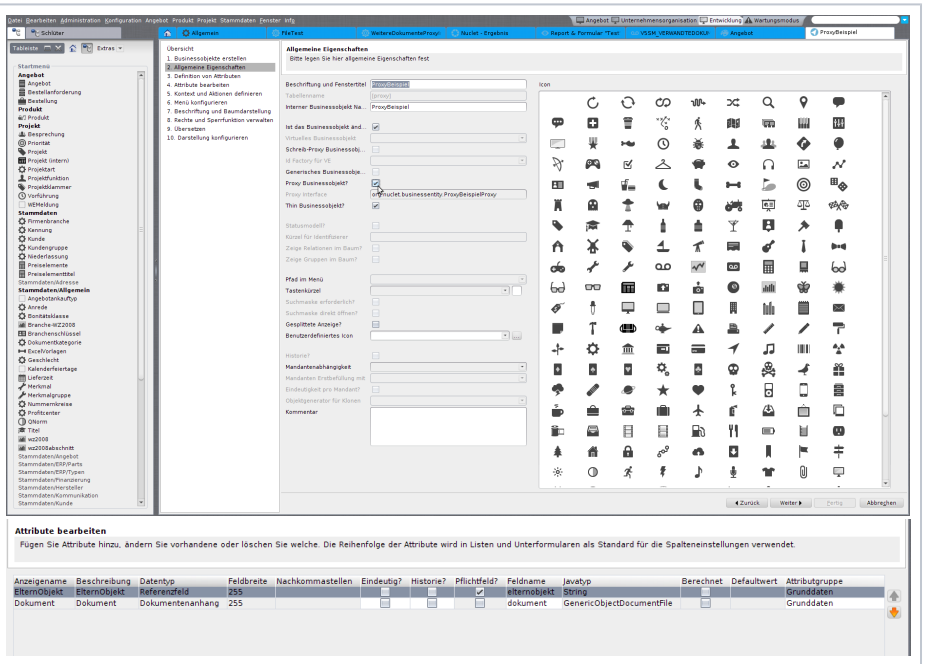
Konfiguration

Erstellen Sie ein Proxy-BO mit Hilfe des BO-Wizards.

Legen Sie hier die gewünschten Felder an, mindestens werden Sie ein Feld für den Dokumentenanhang, sowie eine Referenz auf das **[eltern-objekt]** benötigen.

An dieses **[eltern-objekt]** wird unser Proxy-BO später als Subform angehängt.

Im Zuge dessen wird auch automatisch ein Interface generiert, welches wir später implementieren müssen.

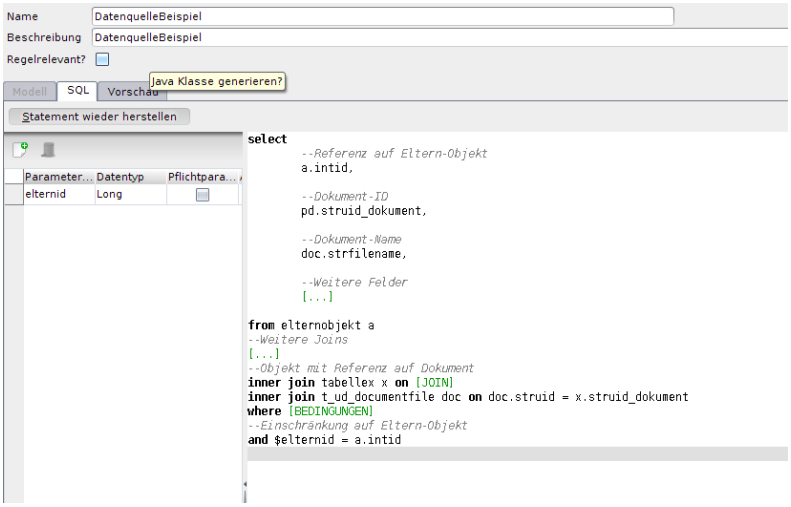


Legen Sie nun eine Datenquelle an, in der Sie die eigentliche View implementieren.

Sie können sich am Beispiel orientieren, entscheiden ist aber vor allem, dass die folgenden vier Felder vorhanden sind:

- *elternid* = Referenz auf das **[eltern-objekt]**
- *dokumentid* = ID des Dokuments, liegt in der Tabelle **t_ud_documentfile**
- *dokumentname* = Name des Dokuments, liegt in der Tabelle **t_ud_documentfile**

Des Weiteren ist es wichtig, dass die Möglichkeit besteht, die Ergebnisdaten über einen Parameter auf das gegebene **[eltern-objekt]** einzuschränken.



Als nächstes muss nun eine Regel erstellt werden, in der das automatisch generierte Interface implementiert wird.

Legen Sie über Regelwerke Regeln (serverseitig) eine neue Regel an - es empfiehlt sich, diese nach dem Interface zu benennen.

Hier müssen nun einige Methoden implementiert werden.

Für unseren Fall reicht es aber aus, lediglich die Methode **getByElternObjekt (java.lang.Long pElternobjektId)** so zu bauen, dass tatsächlich Daten zurückgegeben werden, da nur diese aufgerufen wird, wenn das Proxy-Objekt über das Layout des **[eltern-objekt]** aufgerufen wird.

Alle anderen Methoden können ***null*** zurück geben (sofern ein Rückgabewert erforderlich ist).

Beispiel Interface Implementierung

```
package org.nuclet.[nuclet];

import java.io.File;
import java.io.FileFilter;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.apache.commons.io.filefilter.WildcardFileFilter;
import org.nuclos.api.common.NuclosFile;
import org.nuclos.api.exception.BusinessException;
import org.nuclos.api.provider.DatasourceProvider;
import org.nuclos.api.provider.FileProvider;



public class BeispielObjektProxyImpl implements org.nuclet.[nuclet].
BeispielObjektProxy{
        /*
         * Getter-Methode für das Eltern-Objekt.
         * Wir nutzen den Parameter, um das Ergebnis der Datenquelle
einzuschränken.
         * Es muss immer ein neues Proxy-Objekt erstellt werden, welches
mit den Daten
         * aus der Datenquelle gefüttert wird.
         * Alternativ können die Daten auch von einer anderen Stelle
kommen (Remote-Verbindung, QueryProvider etc.).
         * Für den Dokumentenanhang holen wir uns mit Hilfe von
dokumentid und dokumentname
         * über die Methode getDocument() die Datei aus dem
Installationsverzeichnis.
         * @param pAngebotId = Id des Eltern-Objekts.
         * @return Alle Zeilen des Proxy-BOs, die im Eltern-Objekt
angezeigt werden sollen.
         */
        @Override
        public List<BeispielObjekt> getByElternObjekt(java.lang.Long
pElternobjektId) {
                List<BeispielObjekt> result = new ArrayList<>();
                /*
                 * Setze das Verzeichnis, in dem die Datei zu finden ist.
                 */
            File dir = new File(getHomeDir() + "/data/documents/");
```

```
            try{
                    /*
                     * Setze die Parameter für die Datenquelle und führe
sie aus.
                     */
                        Map<String, Object> params = new HashMap<>();
                        params.put("elternid", pElternobjektId);
                        for (Object[] row :  DatasourceProvider.run
(BeispielDatenquelleDS.class, params).getRows()){
                                    /*
                                     * Erstelle neues Proxy-BO und füttere
es mit Daten.
                                     */
                                    BeispielObjekt b = new BeispielObjekt();
                                    Long elternid = (Long) row[x];
                                    b.setElternId(elternid);
                                    String dokumentid = (String) row[y];
                                    String dokumentname = (String) row[z];
                                    b.setDokument(getDocument(dokumentid,
dokumentname,  dir));
                                    /*
                                     * Setze weitere Felder
                                     */
                                    b.set...(..);

                                    result.add(b);
                            }
                } catch (BusinessException e) {
                    e.printStackTrace();
                }
                    return result;
            }

            /*
             * Ermittelt das Nuclos-Installationsverzeichnis.
             */
        private String getHomeDir() {
            String home = "";
            try {
                    Class<?> clz = Class.forName("org.nuclos.server.common.
NuclosSystemParameters");
                    Method getNuclosHome = clz.getMethod("getNuclosHome");
                            home = ((Path) getNuclosHome.invoke(null)).
toString();
                    } catch (IllegalAccessException |
IllegalArgumentException
                                    | InvocationTargetException |
ClassNotFoundException | NoSuchMethodException | SecurityException e) {
                            e.printStackTrace();
                    }
            return home;
        }

        /*
         * Suche das Dokument im gegebenen Verzeichnis, packe es in eine
NuclosFile und gib
         * dieser den gegeben Namen (andernfalls würden die Dokumente in der
View unter ihrer
         * ID angezeigt werden).
         * @param documentId = ID des Dokuments.
         * @param anzeigename = Gewünschter Anzeigename des Dokuments.
         * @param dir = Verzeichnis, in dem das Dokument gesucht wird.
         * @return Eine NuclosFile oder null, wenn kein passendes Dokument
gefunden wurde.
         */
            private NuclosFile getDocument(String documentId, String
anzeigename, File dir) throws BusinessException {
                    FileFilter fileFilter = new WildcardFileFilter
(documentId + ".*");
                    File[] files = dir.listFiles(fileFilter);
                    if (files.length == 1) {
```

```
                        NuclosFile file = FileProvider.newFile(files[0]);
                        file.setName(anzeigename);
                        return file;
                }
                return null;
        }

        /*
         * Alle anderen Methoden müssen nicht weiter implementiert
werden.
         */

        @Override
        public void setUser(org.nuclos.api.User user) {
        }
        @Override
        public List<BeispielObjekt> getAll() {
                return null;
        }
        @Override
        public List<java.lang.Long> getAllIds() {
                return null;
        }
        @Override
        public BeispielObjekt getById(java.lang.Long id) {
                return null;
        }
        @Override
        public List<BeispielObjekt> getByDokument(org.nuclos.api.UID
pNuclosdocumentfileId) {
                return null;
        }
        @Override
        public void insert(BeispielObjekt pWeitereDokumente) throws org.
nuclos.api.exception.BusinessException {
        }
        @Override
        public void update(BeispielObjekt pWeitereDokumente) throws org.
nuclos.api.exception.BusinessException {
        }
        @Override
        public void delete(java.lang.Long id) throws org.nuclos.api.
exception.BusinessException {
        }
        @Override
        public void commit() {
        }
        @Override
        public void rollback() {
        }
}
```

Zu guter Letzt muss das Proxy-BO noch ins Layout des **[eltern-objekt]** eingefügt werden. Das Vorgehen unterscheidet sich nicht von dem für eine "normale" Subform.	