Nuclet: EZB - Wechselkurse(noch nicht verfügbar)

Nuclet für Währungsumrechnungen und Wechselkursfunktionalität

Name:	Exchange Rates
Package:	org.nuclet.exchangerates
Version:	1.4.0
Datum:	13.09.2017
Nuclos-Kompatibilität:	ab Nuclos 4.19.3

Voraussetzungen

Das Wechselkurs-Nuclet beinhaltet keine Währungsbusinessobjekt mehr. Eine Währungsbusinessobjekt muss im Ziel-Nuclet also existieren oder im Rahmen einer Integration angelegt werden. Zwingend notwendig für diese Währungsbusinessobjekt ist, dass ein Attribut existiert, in dem der dreistellige ISO-4217-Währungscode als Identifizierungsmerkmal gespeichert wird (siehe dazu auch Wikipedia). Weitere Voraussetzungen an die Währungsbusinessobjekte entnehmen Sie bitte dem Abschnitt "Integration" auf dieser Seite.

Es ist möglich und vorgesehen, die Währung aus dem Currency-Nuclet zu nutzen.

Überblick

Kurzbeschreibung

Das Wechselkurs-Nuclet bietet die Möglichkeit

- Wechselkurse zu pflegen
- Währungsbeträge von einer Währung in eine andere zu konvertieren
- Wechselkurse über eine Schnittstelle bei der Europäischen Zentralbank (EZB) täglich aktualisieren zu lassen

Das Wechselkurs-Nuclet war bislang mit dem Währungs-Nuclet ("Currency") gebündelt, die Währungsbusinessobjekte wurde nun von der Wechselkursfunktionalität getrennt.

Das Wechselkurs-Nuclet ist geeignet, mit allen Währungsbusinessobjekten im Zusammenspiel zu funktionieren.

Nuclet-Bestandteile

Das Currency-Nuclet umfasst im Rahmen der .nuclet-Datei

- zwei Businessobjekte (für Wechselkurse und einen Währungsrechner),
- zwei Layouts,
- diverse Java-Regeln (verteilt auf Packages),
- zwei Jobsteuerungen (zur Steuerung des EZB-Schnittstelle einerseits, für Tests andererseits),
- eine Strukturdefinition (für den Import von Stammdaten) und
- eine Nuclet-Abhängigkeit.

Darüberhinaus werden in der ZIP-Datei folgende Komponenten mitgeliefert:

- eine CSV-Datei für Objektimporte

Typ	Name, englisch	Name, deutsch	Kurzbeschreibung
Businessobjekt	Currency Converter	Währungsrechner	
	Exchange Rate	Wechselkurs	
Layout	Currency Converter		Layout für Businessobjekt „Currency Converter“
	Exchange Rate		Layout für die Businessobjekt „Exchange Rate“
Java-Package	org.nuclet.exchangerates.facade		Java-Regeln für Datenbankzugriffe je Businessobjekt
	org.nuclet.exchangerates.job		Java-Regeln zur Steuerung von Jobs („Jobsteuerung“)
	org.nuclet.exchangerates.logic		Geschäftslogik
	org.nuclet.exchangerates.rule		Regeln für Insert-,Update- und Delete-Events
	org.nuclet.exchangerates.test		Java-Regeln zur Steuerung von Tests
	org.nuclet.exchangerates.wrapper		Wrapper-Klassen als Nuclet-Schnittstelle
	org.nuclet.exchangerates.xml		XML-Parserfunktionalität für die EZB-Schnittstelle
Jobsteuerung	Currency Converter Test		Test-Job für Währungskonvertierungen
	ECB Exchange Rates		Fristenjob für die Aktualisierung von Wechselkursen
Strukturdefinition	Currency Converter		Importstruktur für Businessobjekt „Currency Converter“
Nuclet-Abhängigkeit	org.nuclet.Common		allgemeine Helferfunktionalität
Objektimport	Currency_Converter.csv		Stammdatensätze für Businessobjekt „Currency Converter“

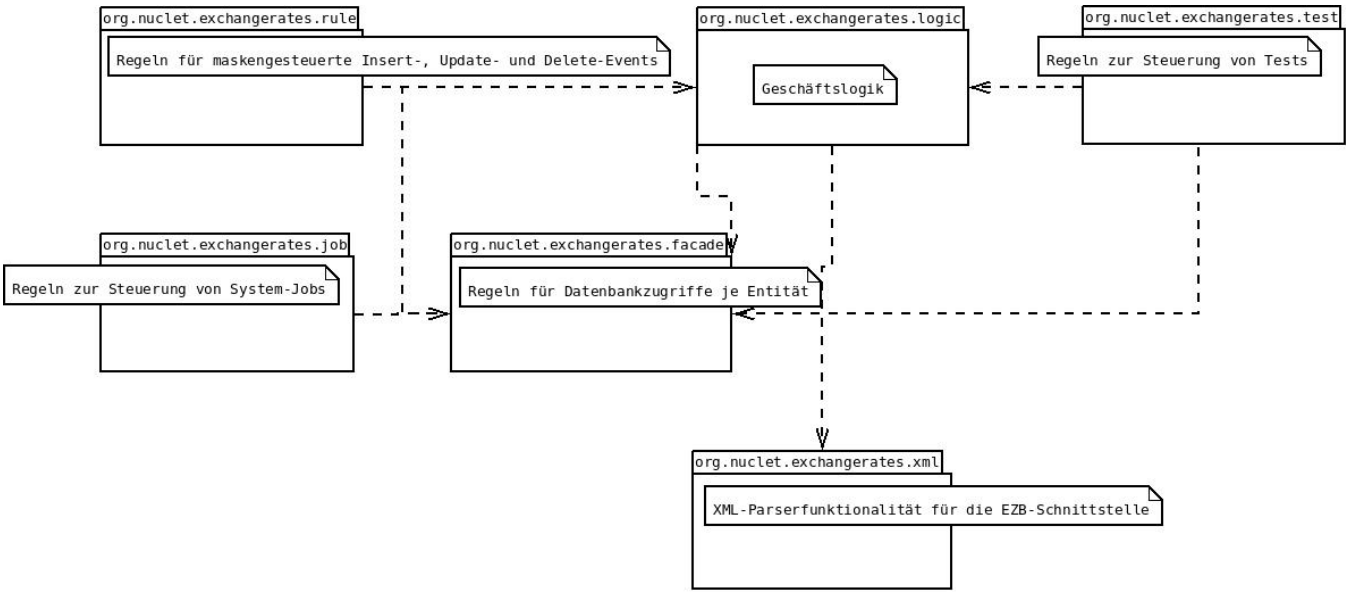
Tabelle 1: Nuclet-Bestandteile

Java-Package-Struktur

Die Java-Regeln sind in sieben Packages unterteilt:

- Regeln für maskengesteuerte Events (org.nuclet.exchangerates.rule)
- Regeln zur Steuerung von System-Jobs (org.nuclet.exchangerates.job)
- Regeln für die Geschäftslogik (org.nuclet.exchangerates.logic)
- Fassaden-Klassen für die Ausführung von Datenbankabfragen je Enität (org.nuclet.exchangerates.facade)
- Regeln zum Einlesen und Verarbeiten von MT940-Dateien (org.nucletexchangerates.xml)
- Regeln für die Ausführung von Tests (org.nuclet.exchangerates.test)
- Wrapper-Klassen als Nuclet-Schnittstelle (org.nuclet.exchangerates.wrapper)

Die Abhängigkeiten der Packages sind in der folgenden Abbildung veranschaulicht.



Das Wechselkurs-Nuclet ist Java-seitig voll funktionsfähig. Hier eine Übersicht über die auf die einzelnen Packages verteilten Java-Klassen:

Java-Package	Java-Klasse	Kurzbeschreibung
org.nucket.exchangerates.facade	AbstractCurrencyFacade	Schnittstellendefinition für Datenbankzugriffe auf ein Währungsbusinessobjekt
	CurrencyFacade	konkrete Ausprägung für Datenbankzugriffe auf ein Währungsbusinessobjekt
	CurrencyFacadeFactory	Factory-Klasse für die CurrencyFacade
	ExchangeRateFacade	Datenbankzugriffe auf das Businessobjekt „Exchange Rate“
org.nuclet.exchangerates.job	EcbExchangeRateImporter	Importer-Klasse für die EZB-Schnittstelle
	EcbExchangeRatesJob	Job für den Aufruf des EZB-Schnittstellenimports
org.nuclet.exchangerates.logic	CurrencyConverterLogic	Logik für Währungskonvertierungen
	CurrencyLogic	Validierungslogik für Währungen
org.nuclet.exchangerates.rule	ConvertCurrencyAmount	Konvertierung in der Währungsrechner-Maske
	ValidateBaseCurrency	Validierungen hinsichtlich der Existenz einer Basis-Währung
	SwapCurrencies	Währungstausch (Basis- und Zielwährung) in der Währungsrechner-Maske
org.nuclet.exchangerates.test	CurrencyConverterTest	Test-Klasse zum Testen von Währungskonvertierungen
	CurrencyConverterTestJob	Job zum aufruf des Tests von Währungskonvertierungen
org.nuclet.exchangerates.wrapper	AbstractCurrencyWrapper	Wrapper-Objekt für Schnittstellendefinition auf Währungsbusinessobjekt
	CurrencyWrapper	konkrete Implementierung des Wrapper-Objektes für Währungen
org.nuclet.exchangerates.xml.sax	EcbExchangeRate	Wrapper-Klasse für den Import über die EZB-Schnittstelle
	EcbExchangeRateHandler	XML-Handler für die EZB-Schnittstelle

Tabelle 2: Java-Package-Struktur

Anwendungsspezifische Anpassungen sollten in den Java-Regeln i.A. nicht notwendig sein.

Integration

Notwendige Schritte zur Integration

Die Integration des Wechselkurs-Nuclets erfolgt in 10 Schritten:

1. Download
2. Nuclet-Import
3. Anlegen eines Währungsbusinessobjekt oder Import des Currency-Nuclets
4. Anpassungen im Businessobjekte
5. Objektimport anlegen
6. Objektimport ausführen
7. Zuordnung von Regeln im Regelmanagement
8. Anpassungen in den Java-Regeln
9. Layout-Anpassungen
10. aktuelle Wechselkurse einlesen

Alle Integrationschritte werden im folgenden im Detail erläutert.

Schritt 1: Download

Download der ZIP-Datei „Exchange_Rates-v1.4.0_4.19.3.zip“ auf der Nuclos-Webpage unter „Nuclos Services“ > „Download“ > „Nuclet Download“. Das ZIP anschließend lokal entpacken.

Schritt 2: Nuclet-Import

Import des MT940-Nuclets unter „Konfiguration“ > „Nuclet Management“ > „Importieren“ in Ihre bestehende Nuclos-Instanz, Auswahl der Datei „Exchange_Rates-v1.4.0-4.19.3.nuclet“

Meldungen beim Import

Beim Import werden die folgenden vier Warnhinweise erscheinen:

- Businessobject Currency Converter references to unknown businessobject Currency. Redirect to dummy businessobject!
- Businessobject Currency Converter references to unknown businessobject Currency. Redirect to dummy businessobject!
- Businessobject Exchange Rate references to unknown businessobject Currency. Redirect to dummy businessobject!
- Businessobject Exchange Rate references to unknown businessobject Currency. Redirect to dummy businessobject!

Der Grund dafür ist: Die Währungsbusinessobjekte als Referenzbusinessobjekt fehlt in den Businessobjekten "Wechselkurs" und "Währungsrechner".

Dazu mehr im Schritt 4.

Schritt 3: Anlegen eines Währungsbusinessobjekt oder Import des Currency-Nuclets

Voraussetzung für das Funktionieren des Wechselkurs-Nuclet ist die Existenz eines Währungsbusinessobjekts mit folgenden Pflichtfeldern:

Attribut	Datentyp	Funktion	Umsetzung im Currency-Nuclet
ISO-4217-Code	Text	Identifizierung der Währung über den dreistelligen Währungscode (siehe Wikipedia)	iso4217Code
Anzahl der Nachkommastellen	Ganzzahl	die Anzahl der Nachkommastellen des Währungssystems bestimmen die Nachkommastellen der Wechselkurse	numberOfDigits
Basiswährung?	Ja/Nein	Definition einer system-eindeutigen Basiswährung (hier: EUR)	isBaseCurrency

Das Währungsbusinessobjekt "Currency" im Currency-Nuclet genügt diesen Voraussetzungen.

Die Attribute müssen nicht so heißen, wie hier angegeben. Eine übereinstimmende Datentypisierung ist hingegen wichtig. Eine Zuordnung der tatsächlich verwendeten Attribute der Währungsbusinessobjekte aus dem Zielnuclet zu den Schnittstellenattributen erfolgt dann im Schritt 8a).

Schritt 4: Anpassungen im Businessobjekt

Über das Businessobjekt muss dann die tatsächlich genutzte Währungsbusinessobjekte als Referenzbusinessobjekte in die Attribute "Basiswährung" und "Zielwährung" der beiden businessobjekten "Wechselkurs" und "Währungsrechner" eingetragen werden.

Schritt 5: Objektimport anlegen

Schritt 5 und 6 sind nur dann notwendig, falls die mitgelieferten Währungen ins Zielsystem importiert werden sollen. Voraussetzung für die Funktion des Nuclets ist jedoch die Existenz zweier Währungen EUR und USD.

Objektimport („Konfiguration“ > „Import & Export“) anlegen zur Strukturdefinition „Currency Converter“. Dazu kann die mitgelieferte CSV-Datei „Currency_Converter.csv“ (zu finden im Unterverzeichnis „data“ der ZIP-Datei) verwendet werden.

Schritt 6: Objektimport ausführen

Wenn der angelegte Objektimport ausgeführt wird, wird ein Währungsrechner-Objekt angelegt.

Achtung: Voraussetzung ist die Existenz der zwei Währungen EUR und USD.

Schritt 7: Zuordnung von Regeln im Regelmanagement

Über das Regelmanagement muss die Java-Regel **ValidateBaseCurrency** der tatsächlich genutzten Währungsbusinessobjekt zugeordnet werden**:**

Quellnuclet	Regeltyp	Regel	Zielnuclet	Businessobjekt
ExchangeRates	Aktualisieren	ValidateBaseCurrency	*das Nuclet, in dem die tatsächlich von Ihnen genutzte Währungsbusinessobjekt liegt*	*ihre eigenes Währungsbusinessobjekt*
ExchangeRates	Anlegen	ValidateBaseCurrency	*das Nuclet, in dem die tatsächlich von Ihnen genutzte Währungsbusinessobjekt liegt*	*ihre eigenes Währungsbusinessobjekt*
ExchangeRates	Löschen	ValidateBaseCurrency	*das Nuclet, in dem die tatsächlich von Ihnen genutzte Währungsbusinessobjekt liegt*	*ihre eigenes Währungsbusinessobjekt*

Schritt 8: Anpassungen in den Java-Regeln

Das Wechselkurs-Nuclet funktioniert mit jedem beliebigen Währungsbusinessobjekt, daher sind bei einer Integration einige Anpassungen in den Java-Regeln durchzuführen:

	Java-Package	Java-Regel	Kurzbeschreibung
a	org.nuclet.exchangerates.wrapper	CurrencyWrapper	Wrapper-Objekt für Währungen, d.h. hier wird eine Schnittstelle zur tatsächlich genutzten Währungsbusinessobjekt definiert
b	org.nuclet.exchangerates.facade	CurrencyFacade	Definition von notwendigen Datenbankzugriffen auf das tatsächlich genutzte Währungsbusinessobjekt
c	org.nuclet.exchangerates.rule	ValidateBaseCurrency	Nutzung des in 8a) definierten Wrapper-Objekts, d.h. das Business-Objekt der Währungsbusinessobjekt aus dem RuleContext wird "gewrappt"

a) CurrencyWrapper

Die Klasse CurrencyWrapper dient als Nuclet-Schnittstelle zum tatsächlich genutzten Währungsbusinessobjekt. Der Methoden und der Konstruktor der Klasse sind mit anwendungsspezifischem Verhalten zu befüllen. Beispiele dazu sind jeweils in Kommentarblöcken angegeben; diese Beispiele sind bei der Integration also an das tatächlich genutzte Währungsbusinessobjekt (bzw. deren BusinessObject-Klasse) anzupassen.

org.nuclet.exchangerates.wrapper.CurrencyWrapper

```
package org.nuclet.exchangerates.wrapper;


import org.nuclos.api.businessobject.BusinessObject;

// @replace
//
// mit eigenem Code zu ersetzen, Beispiel:
//
// import org.nuclet.currency.Currency;


/**
 * Konkrete Wrapper-Klasse für Währungsobjekte
 *
 */
public class CurrencyWrapper extends AbstractCurrencyWrapper
{
    public CurrencyWrapper(final BusinessObject currency)
    {
        // @replace Bitte bei Nuclet-Integration mit eigenem Code ersetzen!
        //
        // Beispiel:
        //
        // if (currency instanceof Currency) {
        //    this.businessObject = currency;
        // }
    }

    /**
     * Liefert die Datenbank-ID des Währungsobjektes.
```

```
     */
    public Long getId()
    {
        return this.businessObject.getId();
    }

    /**
     * Liefert den ISO-4217-Code des Währungsobjektes.
     * @see https://de.wikipedia.org/wiki/ISO_4217
     * @see https://en.wikipedia.org/wiki/ISO_4217
     */
    public String getIso4217Code()
    {
        // @replace Bitte bei Nuclet-Integration mit eigenem Code ersetzen!
        //
        // Beispiel:
        //
        // return ((Currency)this.businessObject).getIso4217Code();

        return null;
    }

    /**
     * Liefert die Anzahl der Nachkommastellen im Währungssystem der Währung
     *
     * @return die Anzahl der Nachkommastellen im Währungssystem der Währung
     */
    public Integer getNumberOfDigits()
    {
        // @replace Bitte bei Nuclet-Integration mit eigenem Code ersetzen!
        //
        // Beispiel:
        //
        // return ((Currency)this.businessObject).getNumberOfDigits();

        return null;
    }

    /**
     * Yields information, if the currency is marked as "base currency"
     *
     * @return true, falls die Währungs als "base currency" markiert ist
     */
    public Boolean getIsBaseCurrency()
    {
        // @replace Bitte bei Nuclet-Integration mit eigenem Code ersetzen!
        //
        // Beispiel:
        //
        // return ((Currency)this.businessObject).getIsBaseCurrency();

        return null;
    }
}
```

b) CurrencyFacade

In der Klasse CurrencyFacade sind die Methoden getBaseCurrencies() und getCurrencyByIso4217Code() mit anwendungsspezifischem Verhalten gefüllt werden. Ein Beispiel dazu ist jeweils in einem Kommentarblock angegeben, dieses Beispiel ist an das tatächlich genutzte Währungsbusinessobjekt (bzw. deren BusinessObject-Klasse) anzupassen.

org.nuclet.exchangerates.facade.CurrencyFacade

```
package org.nuclet.exchangerates.facade;
```

```
import java.util.ArrayList;
import java.util.List;

import org.nuclos.api.businessobject.Query;
import org.nuclos.api.businessobject.QueryOperation;
import org.nuclos.api.businessobject.SearchExpression;
import org.nuclos.api.context.JobContext;
import org.nuclos.api.context.RuleContext;
import org.nuclos.api.exception.BusinessException;
import org.nuclos.api.provider.QueryProvider;
import org.nuclos.api.provider.BusinessObjectProvider;

import org.nuclet.exchangerates.wrapper.AbstractCurrencyWrapper;
import org.nuclet.exchangerates.wrapper.CurrencyWrapper;

// @replace! Bitte bei Nuclet-Integration mit eigenem Code ersetzen!
//
// Beispiel:
//
// import org.nuclet.currency.Currency;


/**
 * Facade for Business Objects of type "Currency"
 *
 */
public class CurrencyFacade extends AbstractCurrencyFacade
{
    public CurrencyFacade(final JobContext jobContext)
    {
        super(jobContext);
    }

    public CurrencyFacade(final RuleContext ruleContext)
    {
        super(ruleContext);
    }

    /**
     * Liefert die Singleton-Instanz dieser Klasse
     *
     */
    public static CurrencyFacade getInstance(final JobContext jobContext)
    {
        return new CurrencyFacade(jobContext);
    }

    public static CurrencyFacade getInstance(final RuleContext ruleContext)
    {
        return new CurrencyFacade(ruleContext);
    }

    /**
     * Fetches all currencies marked as "base currency" from the database and returns
     * a list of wrapped currency objects
     *
     * @return a list of wrapped currency objects
     */
    protected List<AbstractCurrencyWrapper> getBaseCurrencies() throws BusinessException
    {
        // @replace! Bitte bei Nuclet-Integration mit eigenem Code ersetzen!
        //
        // Beispiel:
        //
        // final Query<Currency> queryGetBaseCurrency = QueryProvider.create(Currency.class);
        // queryGetBaseCurrency.where(new SearchExpression(Currency.IsBaseCurrency, Boolean.TRUE,
QueryOperation.EQUALS));
        //
        // final List<Currency> lstBaseCurrencies = QueryProvider.execute(queryGetBaseCurrency);
        // final List<AbstractCurrencyWrapper> lstWrappedBaseCurrencies = new
```

```
ArrayList<AbstractCurrencyWrapper>();
        //
        // for (final Currency currency : lstBaseCurrencies) {
        //     lstWrappedBaseCurrencies.add(new CurrencyWrapper(currency));
        // }

        // return lstWrappedBaseCurrencies;

        return null;
    }

    /**
     * Fetches the currency which determined by a given ISO 4217 code from the database
     *
     * @param strIso4217Code the ISO 4217 code of the currency to be found
     *
     * @return the currency which is marked as "base currency" from the database
     * @throws BusinessException, if more than one currency-objects are found
     */
    public AbstractCurrencyWrapper getCurrencyByIso4217Code(final String strIso4217Code)
    {
        // @replace! Bitte bei Nuclet-Integration mit eigenem Code ersetzen!
        //
        // Beispiel:
        //
        //  final Query<Currency> queryGetCurrency = QueryProvider.create(Currency.class);
        //  queryGetCurrency.where(Currency.Iso4217Code.eq(strIso4217Code));
        //
        //  final List<Currency> listCurrency = QueryProvider.execute(queryGetCurrency);
        //
        //  if (listCurrency != null && listCurrency.size() > 0) {
        //     return new CurrencyWrapper(listCurrency.get(0));
        //  } else {
        //      return null;
        //  }

        return null;
    }



}
```

c) ValidateBaseCurrency

In der Regel ValidateBaseCurrency ist die BusinessObject-Klasse des tatsächlich genutzte Währungsobjektes einzutragen.

org.nuclet.exchangerates.rule.ValidateBaseCurrency

```
package org.nuclet.exchangerates.rule;

import org.nuclos.api.rule.DeleteRule;
import org.nuclos.api.rule.InsertRule;
import org.nuclos.api.rule.UpdateRule;
import org.nuclos.api.context.DeleteContext;
import org.nuclos.api.context.InsertContext;
import org.nuclos.api.context.RuleContext;
import org.nuclos.api.context.UpdateContext;
import org.nuclos.api.annotation.Rule;
import org.nuclos.api.exception.BusinessException;
import org.nuclos.api.provider.BusinessObjectProvider;
import org.nuclos.api.provider.QueryProvider;

import org.nuclet.common.rule.AbstractRule;
```

```
import org.nuclet.exchangerates.logic.CurrencyLogic;
import org.nuclet.exchangerates.wrapper.AbstractCurrencyWrapper;
import org.nuclet.exchangerates.wrapper.CurrencyWrapper;

// @replace! Bitte bei Nuclet-Integration mit eigenem Code ersetzen!
//
// Beispiel:
//
// import org.nuclet.currency.Currency;

/**
 * @name ValidateBaseCurrency
 * @description Checks if exactly one base currency is declared
 * @usage
 * @change
 *
 *
 */
@Rule(name="ValidateBaseCurrency", description="Checks if exactly one base currency is declared")
public class ValidateBaseCurrency extends AbstractRule implements InsertRule, UpdateRule, DeleteRule
{
    public void insert(InsertContext context) throws BusinessException
    {
        // @replace Bitte bei Nuclet-Integration mit eigenem Code ersetzen!
        //
        // Beispiel:
        //
        // validateBaseCurrency(context, new CurrencyWrapper(context.getBusinessObject(Currency.class)));
    }

    public void update(UpdateContext context) throws BusinessException
    {
        // @replace Bitte bei Nuclet-Integration mit eigenem Code ersetzen!
        //
        // Beispiel:
        //
        // validateBaseCurrency(context, new CurrencyWrapper(context.getBusinessObject(Currency.
class)));
    }

    public void delete(DeleteContext context) throws BusinessException
    {
        // @replace Bitte bei Nuclet-Integration mit eigenem Code ersetzen!
        //
        // Beispiel:
        //
        // validateBaseCurrency(context, new CurrencyWrapper(context.getBusinessObject(Currency.
class)));
    }


    private void validateBaseCurrency(final RuleContext context,
        final AbstractCurrencyWrapper currency) throws BusinessException
    {
        initialize(context);

        final CurrencyLogic currencyLogic = new CurrencyLogic(context);
        currencyLogic.validateCurrency(currency);
    }
}
```

Schritt 9: Layout-Anpassungen

Im Layout "Currency" können Sie die Wechselkurse als Unterformular einbinden.

Schritt 10: aktuelle Wechselkurse einlesen

Über den Job **ECB Exchange Rates** lassen sich nun die aktuell gültigen Wecheselkurse einlesen. Es werden nur diejenigen Währungen berücksichtigt, die im System vorliegen.

Nutzung

Einrichten der Jobsteuerung

1. Öffnen der Jobsteuerung „ECB Exchange Rates“ unter „Administration“ > „Jobsteuerung“
2. Konfiguration von Startdatum, Startzeit, Intervall, E-Mail-Benachrichtigung und Optionen zum Löschen des Log-Infos.
3. Jobsteuerung aktivieren, falls der Job im konfigurierten Rhythmus automasiert ausgeführt werden soll. Alternativ lässe sich der Job über die Jobsteuerung auch manuell ausführen.

Betrieb

1. Die importierten Daten werden in der businessobjekt für Wechselkursee („Exchange Rate“) gespeichert.
2. Die Masken für Währungen und Wechselkurse sind unter dem Menüpunkt „Wechselkurse“ zu finden. Dies ist über das Businessobjekt jederzeit zu ändern.
3. Der Währungsrechner ist ebenfalls unter dem Menüpunkt „Wechselkurse“ zu finden. Sollten Sie ihn nicht nutzen wollen, dann können Sie ihn über das Businessobjekt aus dem Menü entfernen.
4. Das Nuclet ist zur Zeit so eingestellt, dass alle Wechselkurse in der Basiswährung des System angezeigt werden. Da die Aktualisierung über die EZB-Schnittstelle läuft, ist der Euro (EUR) als Basiswährung hinterlegt. Sie können das Nuclet auch leicht so einstellen, dass in jeder Währung der Wechselkurs zum Euro angezeigt wird: Dazu öffnen Sie das Layout „Currency“ und tauschen den eingetragenen Wert im Feld „Fremdschlüssel“ (Unterformular „Exchange Rates“) von „baseCurrency“ auf „counterCurrency“.

Einrichten von Benutzerrechten (optional)

Zugriffsrechte für existierende Benutzergruppen für Layouts, Businessobjekte, Attribute und Attributgruppen können in Nuclos wie üblich über drei Wege festgelegt werden (siehe dazu http://wiki.nuclos.de):

1. allemeine Einstellungen für Businessobjekte unter „Administration“ > „Benutzergruppen“
2. statusabhängige Einstellungen für Attribute/Attributgruppen unter „Konfiguration“ > „Statusmodell“
3. datensatzspezifische Einstellungen unter „Konfiguration“ > „Datenquellen“ > „Datensatzfreigabe“

Tests

- Für einfache Tests von Währungskonvertierungen steht die Währungsrechner-Maske zur Verfügung.
- Für erweiterte Tests von Währungskonvertierungen nutzen Sie bitte die Klasse org.nuclet.currency.test.CurrencyConverterTest. Diese können Sie auch gerne auf Ihre eigenen Bedürfnisse anpassen und erweitern. Die Klasse wird vom org.nuclet.currency.test. CurrencyConverterTestJob aufgerufen, und dieser lässt sich über die Jobsteuerung „Currency Converter Test“ bei Bedarf (am besten manuell) ausführen.

Versionen

Version	Datum	Typ	Änderungen
1.0.0	12.08.2013	initiale Version	Trennung von Wechselkursfunktionalität und Währungsbusinessobjekt aus Currency-Nuclet
1.0.1	08.10.2013	Bugfixes und Erweiterungen	siehe Release Notes
1.1.0	08.01.2014	Umstellung auf Nuclos 4.0	Umstellung auf Nuclos 4.0
1.2.0	25.03.2017	Umstellung auf Nuclos 4.13	Umstellung auf Nuclos 4.13
1.3.0	08.08.2017	Umstellung auf Nuclos 4.18	Umstellung auf Nuclos 4.18
1.4.0	13.09.2017	Umstellung auf Nuclos 4.19	Umstellung auf Nuclos 4.19