Festlegungen für Nuclos API Entwicklung

Diverse Festlegungen

- Sind Collections betroffen, sollten die Methodennamen ein "...All()" im Namen haben, z.B. insert(BusinessObject) - insertAll(Collection <BusinessObject>)
- Standardbenennung für Löschmethoden: delete (nicht remove, drop, etc.)
- Standardbenennung für Einfügemethoden: insert (nicht new, create, etc.)
- Standardbenennung für Aktualisierungsmethoden: update (nicht modify, etc.)
- Thema Id's:
  - Id statt ID (z.B. getId())
  - Methoden die allgemein eine Id zurückgeben sollten auch auf "...Id()" enden
  - Id's müssen immer vom Datentyp Long/UID sein
- Zusammenspiel Service - Provider...

Fehlerbehandlung Exceptionshandling

Bitte immer folgendes beachten:

1) Keine Exception still abfangen. Es muss immer wenigstens eine Warnung ins Log geschrieben werden.

2) Auf dem Dispatch Thread keine RuntimeException (z.b. CommonFatalException) werfen. Das bringt die GUI zum Komplett Absturz und Einfrieren.

3) Möglichst nicht allgemeine Exceptions "Exception/RuntimeException" catchen. Manchmal lässt sich das nicht ohne viel Aufwand vermeiden, das sollte aber in jedem Fall minimal gehalten werden. Am Besten gar nicht machen.

In diesem Zusammenhang bitte sog. lange Methoden Ketten vermeiden. Solche Ketten sind ganz schlecht zu debuggen und bei Stacktraces hat man keine Ahnung, was genau passiert ist. vgl. Java-Programmierstil Richtlinien für Nuclos

Datentypen für Ganzzahlen und Fließkommazahlen

Bei der Entwicklung sollte darauf geachtet werden, dass für Zahlen (Number) nur noch folgende zwei Datentypen verwendet werden:

- **Long für Ganzzahlen**
- **BigDecimal für Fließkommazahlen**

Long liegt zur Zeit der 64-bit Rechner und 64-bit Systeme nahe, da diese gegenüber dem 32--bit-Integer keinerlei Nachteile haben und einen wesentlich größeren Zahlenraum abdecken. 32-bit Integer sollte, wenn überhaupt, nur noch ausschließlich für Schleifenvariablen (i,j, etc..) verwendet werden. Rückgabewerte, Keys, Daten und alle Zahlen mit denen gerechnet wird, sollten immer Long sein.

BigDecimal ist unbedingt Double zu bevorzugen. Double ist von der Genauigkeit limitiert und kann bei Rundungen von Zwischenergebnissen für erhebliche Probleme sorgen. BigDecimal benötigt zwar mehr Ressourcen als Double, das hat aber für die typischen kaufmännischen Berechnungen in Nuclos kein merklichen Auswirkungen.

Codeverdoppelung vermeiden

Blindes Copy & Paste von Code ist ein No-Go. Die Wartung ist eine Katastrophe, muss man was ändern, muss man erst alle Stellen zusammensuchen. Der Code bläht sich völlig unnötig auf. Nicht umsonst werden “Code duplicates” von IntelliJ angemeckert. Letztens wurde erst 20-fach kopierter Code aus den Sourcen entfernt.
Bitte Code-Verdoppelung in jedem Fall vermeiden und zwar:

1) Verwendung identischer Methoden in einer abstrakten Überklasse.
2) Falls 1) nicht möglich ist: Auslagerung in eine statische Methode in einer Utils-Klasse.
3) Falls Client und Server gleiches machen wollen: Auslagerung in den Common-Block.

Gerade gibt es einen Fall, wo der Installer und Kern (zwei unabhängige Projekt) gleiches machen. In dem Fall lässt es sich natürlich kaum vermeiden. Das ist die einzige Ausnahme.