

QueryProvider (Beispiele)

Imports:

```
import org.nuclos.api.provider.QueryProvider;
```

```
import org.nuclos.api.businessobject.Query;
```

Erstellen einer Datenbankabfrage (Query)

Methode	Beispiel
create	<p>Allgemein:</p> <p>Mit Hilfe dieser Methode kann ein typisiertes Query-Object angelegt werden, mit dessen Hilfe Datenbankabfragen ausgeführt werden können. Dabei wird auf eine abstrakte Query-Language zurückgegriffen. Abfragen mit SQL-Syntax sind nicht möglich.</p> <pre data-bbox="261 464 1484 688">boolean sortAscending = true; Query<Auftrag> queryAuftrag = QueryProvider.create(Auftrag.class); queryAuftrag.where(Auftrag.Auftragsnr.notNull()) .and(Auftrag.Bestellwert.Gt(BigDecimal.ZERO)) .orderBy(Auftrag.Auftragsnr, sortAscending);</pre> <p>Folgende Bedingungen können verwendet werden:</p> <ul data-bbox="280 758 509 852" style="list-style-type: none">• eq(value): gleich• isNull(): ist ein Nullwert• neq(value): ungleich• notNull(): kein Nullwert <p>Und für Numerische Attribute:</p> <ul data-bbox="280 919 607 1014" style="list-style-type: none">• Gt(value): größer als• Gte(value): Größer als oder gleich• Lt(value): kleiner als• Lte(value): kleiner als oder gleich <p>Weitere Informationen finden sich in folgenden Klassen (unter api.nuclos.de):</p> <p><i>org.nuclos.api.businessobject.attribute.Attribute<T></i></p> <p><i>org.nuclos.api.businessobject.attribute.NumericAttribute<T></i></p> <p>Suchen von Einträgen mit einem bestimmten Status</p> <pre data-bbox="261 1262 1484 1356">Query<Bestellung> qry = QueryProvider.create(Bestellung.class); qry.where(Bestellung.NuclosStateId.eq(BestellungSM.State_60.getId()));</pre> <p>Welchen Status ein BusinessObject einnehmen kann, hängt von dem zugewiesenen Statusmodel ab. Dazu gibt es für jedes Statusmodel eine entsprechende Klasse, hier "BestellungSM", die alle Status als Konstanten beinhaltet.</p>
execute	<p>Diese Funktion führt eine Query aus und gibt eine typisierte Liste als Ergebnis zurück.</p> <pre data-bbox="261 1535 1484 1692">List<Auftrag> results = QueryProvider.execute(queryAuftrag); for (Auftrag a :results) { BigDecimal bestellwert = a.getBestellwert(); }</pre>

Erstellen einer Unterabfrage

Methode	Beschreibung
---------	--------------

exist

Mit Hilfe der exist-Methode kann eine Subquery in eine äußere Query eingebunden werden. Die Verknüpfung von Query und Subquery findet ausschließlich über (Fremd)schlüssel statt.

Hier ein Beispiel. Ziel der Abfrage ist die Ermittlung einer Liste von Bestellungen, für die Kunden mit vollständigen Zahlungsbedingungen hinterlegt sind:

```
// Alle Kunden, fuer die Zahlungsbedingungen hinterlegt wurden
Query<Kunde> qryKunden = QueryProvider.create(Kunde.class);
qryKunden.where(Kunde.Zahlungsbedingung.notNull());

// Nun brauchen wir alle Bestellungen dieser Kunden
Query<Bestellung> queryBestellungen = QueryProvider.create(Bestellung.class);
queryBestellungen.where(Bestellung.Bestellnr.notNull())
    .exist(qryKunden, Bestellung.KundeId)
    .orderBy(Bestellung.Bestellnr, true);
List<Bestellung> results = QueryProvider.execute(queryBestellungen);
```

Soll nicht der PrimaryKey als Referenzfeld der Subquery verwendet werden, sondern ein anderer ForeignKey, muss dieser beim Aufruf der exist()-Methode mit angegeben werden:

```
Auftrag a = context.getBusinessObject(Auftrag.class);

// Subquery zur Ermittlung der Produktgruppe, die im Auftrag angegeben wurde
Query queryPg = QueryProvider.getById(Produktgruppe.class, a.getProduktgruppeId());

// Hauptabfrage des Bestands aller Produkte, die in der Produktgruppe hinterlegt wurden
Query queryBestand = QueryProvider.create(Bestand.class);
queryBestand.exist(queryPg, Bestand.Produkt, Produktgruppe.ProduktId); // Produktgruppe.ProduktId
ist Fremdschlüssel in Produktgruppe

List<Bestand> lstResults = QueryProvider.execute(queryBestand);

for (Bestand b : lstResults) {
    // ...
}
```

Neu (Ab 4.34.0 und 4.33.6: Soll eine echte Subform-Suche ausgeführt werden, d.h. es werden Hauptdatensätze mit einer Subform-Bedingung gesucht, dann gilt folgendes Beispiel:

```
// Alle Bestellungen, deren Betrag größer als 1000 ist.
Query<Bestellung> queryBestellungen = QueryProvider.create(Bestellung.class);
queryBestellungen.where(Bestellung.Betrag.Gte(1000));

// Nun brauchen wir alle Kunden die mindestens eine dieser Bestellungen haben.
Query<Kunde> queryKunden = QueryProvider.create(Kunde.class);
queryKunden.exist(queryBestellungen, null, Bestellung.KundeId); // Die Null im zweiten Argument
kennzeichnet die Subform-Suche

List<Kunde> results = QueryProvider.execute(queryKunden);
```

Zu beachten ist, dass die Subquery selbst nicht vom QueryProvider ausgeführt werden muss. Sie wird in ihrer fertigen "Struktur" der äußeren Query übergeben.

Auslesen eines einzelnen Eintrages

Methoden	Beispiel
----------	----------

getById	<p>Mit Hilfe dieser Methode kann ein einzelnes BusinessObject anhand des Typs und der Id ausgelesen werden.</p> <pre style="border: 1px solid black; padding: 5px;"> // die Variable 'auftrag' stellt ein BusinessObject des Typs 'Auftrag' dar und besitzt eine Referenz auf ein Kundenobjekt Kunde k = QueryProvider.getById(Kunde.class, auftrag.getKundeId()); </pre>
----------------	---

Suche anhand von Aktionen und Status

Methode	Beispiel
getState	<p>Mit Hilfe dieser Methode können Datenbankeinträge in Form von BusinessObjekten ermittelt werden, die einen bestimmten Status besitzen. Für die Suche können mehrere Status angegeben werden, zwingend erforderlich ist aber nur einer.</p> <p>Die Status befinden sich in den Statusmodell-Klassen.</p> <pre style="border: 1px solid black; padding: 5px;"> // Suche nach allen Bestellungen, die sich im Status 50 (storniert) oder Status 60 (beendet) befinden List<Bestellung> results = QueryProvider.getState(Bestellung.class, BestellungSM.State_50, BestellungSM.State_60); context.log("Anzahl der abgeschlossenen Bestellungen: " + results.size()); // Suche nach allen Bestellungen, die sich im Status 50 (storniert) befinden List<Bestellung> results = QueryProvider.getState(Bestellung.class, BestellungSM.State_50); context.log("Anzahl der stornierten Bestellungen: " + results.size()); </pre>
getProcess	<p>Mit Hilfe dieser Methode können Datenbankeinträge in Form von BusinessObjekten ermittelt werden, die einer oder mehreren Aktionen angehören. Eine Aktion ist immer einem Businessobjekt zugewiesen, weshalb sie im Funktionsaufruf nicht extra angegeben werden muss. Aktionen werden in Nuclos konfiguriert und aufgrund ihrer Zugehörigkeit zum Businessobjekt in den BusinessObjekten als Konstanten hinterlegt, z.B. <i>Auftrag.Sonderauftrag</i> oder <i>Auftrag.Normalauftrag</i></p> <pre style="border: 1px solid black; padding: 5px;"> // Liste aller Sonderaufträge List<Auftrag> results = QueryProvider.getByProcess(Auftrag.Sonderauftrag); context.log("Anzahl der Sonderaufträge: " + results.size()); </pre> <p>Anmerkung:</p> <p>Aktionen können in den Businessobjekten direkt gesetzt werden. Wichtig dabei ist, dass aufgrund der Typsicherheit einem Businessobjekt nur die Aktionen zugewiesen werden können, die auch zur entsprechenden Businessobjekt gehören.</p> <pre style="border: 1px solid black; padding: 5px;"> // Hier wird ein bestimmter Auftrag einer Aktion zugewiesen Auftrag a = QueryProvider.getById(Auftrag.class, 40297631L); a.setNuclosProcess(Auftrag.Sonderauftrag); BusinessObjectProvider.update(a); </pre>