# QueryProvider (Beispiele)

# Imports:

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

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

# Erstellen einer Datenbankabfrage (Query)

| Methode | Beispiel |
|---|---|
| create | **Allgemein:**<br><br>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.<br><br>`boolean sortAscending = true;`<br><br>`Query<Auftrag> queryAuftrag = QueryProvider.create(Auftrag.class);`<br><br>`queryAuftrag.where(Auftrag.Auftragsnr.notNull())`<br>`.and(Auftrag.Bestellwert.Gt(BigDecimal.ZERO))`<br>`.orderBy(Auftrag.Auftragsnr, sortAscending);`<br><br>Folgende Bedingungen können verwendet werden:<br><br>• eq(value): gleich<br>• isNull(): ist ein Nullwert<br>• neq(value): ungleich<br>• notNull(): kein Nullwert<br><br>Und für Numerische Attribute:<br><br>• Gt(value): größer als<br>• Gte(value): Größer als oder gleich<br>• Lt(value): kleiner als<br>• Lte(value): kleiner als oder gleich<br><br>Weitere Informationen finden sich in folgenden Klassen (unter api.nuclos.de):<br><br>*org.nuclos.api.businessobject.attribute.Attribute<T>*<br><br>*org.nuclos.api.businessobject.attribute.NumericAttribute<T>*<br><br>**Suchen von Einträgen mit einem bestimmten Status**<br><br>`Query<Bestellung> qry = QueryProvider.create(Bestellung.class);`<br>`qry.where(Bestellung.NuclosStateId.eq(BestellungSM.State_60.getId()));`<br><br>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. |
| execute | Diese Funktion führt eine Query aus und gibt eine typisierte Liste als Ergebnis zurück.<br><br>`List<Auftrag> results = QueryProvider.execute(queryAuftrag);`<br>`for (Auftrag a :results) {`<br>`BigDecimal bestellwert = a.getBestellwert();`<br>`}` |

# 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.<br><br>Hier ein Beispiel. Ziel der Abfrage ist die Ermittlung einer Liste von Bestellungen, für die Kunden mit vollständigen Zahlungsbedingungen hinterlegt sind:<br><br>`// Alle Kunden, fuer die Zahlungsbedingungen hinterlegt wurden`<br>`Query<Kunde> qryKunden = QueryProvider.create(Kunde.class);`<br>`qryKunden.where(Kunde.Zahlungsbedingung.notNull());`<br><br>`// Nun brauchen wir alle Bestellungen dieser Kunden`<br>`Query<Bestellung> queryBestellungen = QueryProvider.create(Bestellung.class);`<br>`queryBestellungen.where(Bestellung.Bestellnr.notNull())`<br>`.exist(qryKunden, Bestellung.KundeId)`<br>`.orderBy(Bestellung.Bestellnr, true);`<br>`List<Bestellung> results = QueryProvider.execute(queryBestellungen);`<br><br>Soll nicht der PrimaryKey als Referenzfeld der Subquery verwendet werden, sondern ein anderer ForeignKey, muss dieser beim Aufruf der exist()-Methode mit angegeben werden:<br><br>`Auftrag a = context.getBusinessObject(Auftrag.class);`<br><br>`// Subquery zur Ermittlung der Produktgruppe, die im Auftrag angegeben wurde`<br>`Produktgruppe produktgruppe = Produktgruppe.get(a.getProduktgruppeId());`<br><br>`// Hauptabfrage des Bestands aller Produkte, die in der Produktgruppe hinterlegt wurden`<br>`Query queryBestand = QueryProvider.create(Bestand.class);`<br>`queryBestand.exist(queryPg, Bestand.Produkt, Produktgruppe.ProduktId); // Produktgruppe.ProduktId ist Fremdschlüssel in Produktgruppe`<br><br>`List<Bestand> lstResults = QueryProvider.execute(queryBestand);`<br><br>`for (Bestand b : lstResults) {`<br>`// ...`<br>`}`<br><br>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:<br><br>`// Alle Bestellungen, deren Betrag größer als 1000 ist.`<br>`Query<Bestellung> queryBestellungen = QueryProvider.create(Bestellung.class);`<br>`queryBestellungen.where(Bestellung.Betrag.Gte(1000));`<br><br>`// Nun brauchen wir alle Kunden die mindestens eine dieser Bestellungen haben.`<br>`Query<Kunde> queryKunden = QueryProvider.create(Kunde.class);`<br>`queryKunden.exist(queryBestellungen, null, Bestellung.KundeId); // Die Null im zweiten Argument kennzeichnet die Subform-Suche`<br><br>`List<Kunde> results = QueryProvider.execute(queryKunden);`<br><br>Zu beachten ist, dass die Subquery selbst nicht vom QueryProvider ausgeführt werden muss. Sie wird in ihrer fertigen "Strduktur" der äußeren Query übergeben. |

# Auslesen eines einzelnen Eintrages

| Methode | Beispiel |
|---|---|

| | |
|---|---|
| get | Mit Hilfe dieser Methode kann ein einzelnes BusinessObject anhand der Id ausgelesen werden.<br><br>`import org.nuclos.api.exception.BusinessException;`<br>`public class UtilsNeu {`<br><br>`    public static void workAuftrag(Long id1, Long id2) throws BusinessException {`<br>`        Auftrag auftrag = Auftrag.get(id1);`<br>`         }`<br>`}` |

# Suche anhand von Aktionen und Status

| Methode | Beispiel |
|---|---|
| getByState | Mit Hilfe dieser Methode können Datenbankeinträge in Form von BusinessObjekten ermittelt werden, die einen bestimmten Status besitzen.<br>Für die Suche können mehrere Status angegeben werden, zwingend erforderlich ist aber nur einer.<br><br>Die Status befinden sich in den Statusmodell-Klassen.<br><br>`// Suche nach allen Bestellungen, die sich im Status 50 (storniert) oder Status 60 (beendet) befinden`<br>`List<Bestellung> results = QueryProvider.getByState(Bestellung.class, BestellungSM.State_50, BestellungSM.State_60);`<br>`context.log("Anzahl der abgeschlossenen Bestellungen: " +  results.size());`<br><br>`// Suche nach allen Bestellungen, die sich im Status 50 (storniert) befinden`<br>`List<Bestellung> results = QueryProvider.getByState(Bestellung.class, BestellungSM.State_50);`<br>`context.log("Anzahl der stornierten Bestellungen: " +  results.size());` |
| getByProcess | 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. *Auftrag.Sonderauftrag* oder *Auftrag.Normalauftrag*<br><br>`// Liste aller Sonderaufträge`<br>`List<Auftrag> results = QueryProvider.getByProcess(Auftrag.Sonderauftrag);`<br>`context.log("Anzahl der Sonderaufträge: " + results.size());`<br><br>Anmerkung:<br><br>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.<br><br>`// Hier wird ein bestimmter Auftrag einer Aktion zugewiesen`<br>`Auftrag a = Auftrag.get(40297631L);`<br>`a.setNuclosProcess(Auftrag.Sonderauftrag);`<br>`a.save();` |