

QueryProvider

🌐 Sprache: **Deutsch** · [English](#)

QueryProvider

Typisierte Datenabfragen in Regeln – `create/where/and/orderBy/execute`, `getByState`, `getByProcess`, Subqueries.

REFERENZ

ENTWICKLER

STAND: JUL 2026

GILT FUER NUCLOS 4.2026.X

Auf dieser Seite

- [Wozu dient der QueryProvider?](#)
- [create\(type\)](#)
- [execute\(query\)](#)
- [get\(id\)](#)
- [getByState / getByProcess](#)
- [Bedingungen & Verknüpfungen](#)
- [Ausführen & iterieren](#)
- [Unterabfragen \(exist\)](#)
- [Verwandte Seiten](#)

Wozu dient der QueryProvider?

Der QueryProvider lädt Daten aus Nuclos und stellt sie als Businessobjekt oder Liste in Regeln bereit. Statt SQL wird eine **typisierte, abstrakte Query-Sprache** verwendet.



Logisch gelöschte Einträge

Ohne `NuclosLogicalDeleted` werden logisch gelöschte Einträge **nicht** berücksichtigt. Nur `NuclosLogicalDeleted.eq(Boolean.TRUE)` liefert gelöschte Einträge.

create(type)

Legt ein typisiertes Query-Objekt für ein Businessobjekt an.

```
public static <T extends BusinessObject> Query<T> create(Class<T> type);
```

execute(query)

Führt die Query aus und liefert eine typisierte, nie null-Liste (leer bei keinem Treffer).

```
public static <T extends BusinessObject> List<T> execute(Query<T> query);
```

get(id)

Liest einen konkreten Eintrag über Typ und Id; null, wenn nichts gefunden.

getByState / getByProcess

Suchen Einträge nach **Status** bzw. nach zugewiesener **Aktion** (Prozess). Mindestens ein Status/eine Aktion ist erforderlich.

Bedingungen & Verknüpfungen

Auf Feldern stehen passende Vergleichsoperatoren bereit: eq, neq, isNull, notNull (alle Typen) sowie Gt, Gte, Lt, Lte (numerisch). where()/and()/orderBy() geben die Query zurück und lassen sich stapeln. Ein or() existiert nur auf *SearchExpression*-Ebene, wodurch sich Blöcke verschachteln lassen.

```
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);
```

Ausführen & iterieren

```
List<Auftrag> results = QueryProvider.execute(queryAuftrag);
for (Auftrag a :results) {
    BigDecimal bestellwert = a.getBestellwert();
}
```

Unterabfragen (exist)

Mit exist(subQuery, ...) lassen sich Subqueries über (Fremd-)Schlüssel einbinden; die Subquery selbst wird nicht separat ausgeführt.

```
// 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);
```

Verwandte Seiten



BusinessObjectProvider

BOs schreiben.

[Öffnen](#)



DatasourceProvider

Datenquellen ausführen.

[Öffnen](#)



GenerationProvider

Arbeitsschritte.

[Öffnen](#)