DatasourceProvider

Sprache: **Deutsch** · English

DatasourceProvider

> **Datenquellen aus Regeln ausführen – run(), DatasourceResult, rows/columns und Datenbankfunktionen.**

REFERENZ | ENTWICKLER | STAND: JUL 2026 | GILT FUER NUCLOS 4.2026.X

Auf dieser Seite

Wozu dient der DatasourceProvider?

Der `DatasourceProvider` führt eine **Datenquelle** (aus Report und Formular) programmatisch aus.

run(datasourceClass[, params])

Führt die Datenquelle aus und liefert ein `DatasourceResult` mit Zeilen (`rows`) und Spalten (`columns`). Parameter werden optional als `Map` übergeben.

```
public DatasourceResult run(Class<? extends Datasource> datasourceClass) throws BusinessException;
public DatasourceResult run(Class<? extends Datasource> datasourceClass, Map<String, Object> params) throws
BusinessException;
```

> **Oracle**
>
> Unter Oracle funktioniert dies nur für Datenquellen, die **keine Daten schreiben** – innerhalb eines SELECT dürfen dort keine Daten verändert werden.

Ergebnis auswerten

`rows` ist eine Liste von `Object[]`; über `getColumns()` erhält man Name und Datentyp je Spalte und kann typgerecht casten.

```
public class AuftragAbschliessen implements UpdateFinalRule {
    public void updateFinal(UpdateContext context) throws BusinessException {
        // Als Parameter geben wir die ID mit
        Map<String, Object> map = new HashMap<String, Object>();
        map.put("intid", er.getId().intValue());

                DatasourceResult run = DatasourceProvider.run(AuftragDS.class, map);

        for (Object[] rowWithColumns : run.getRows()) {
            for (int idx=0; idx < run.getColumns().size(); idx++) {
                DatasourceColumn datasourceColumn = run.getColumns().get(idx);
                ctx.log("Feld: " + datasourceColumn.getName() + " hat den Wert: " +
                        run.getColumns().get(idx).getType().cast(rowWithColumns[idx]).toString());
            }
        }
}
```

Datenbankfunktion aufrufen

Der Provider ersetzt das alte `callDBFunction`: Die Funktion wird in einer Datenquelle gekapselt und indirekt aufgerufen.

```
SELECT CA_MYDBFUNCTION('$parameter')::numeric(9,2) "result"
```

```
package org.nuclet.test;

import org.nuclos.api.exception.BusinessException;
import org.nuclos.api.provider.DatasourceProvider;
import org.nuclos.api.datasource.DatasourceResult;

@Rule(name="Test", description="Test")
public class Test {
    public static Double callFunction() throws BusinessException {
        DatasourceResult ds = DatasourceProvider.run(MyDatasourceDS.class);
        Double result = (Double)(ds.getRows().iterator().next()[0]);
        return result;
    }
}
```

Verwandte Seiten

Datenquellen

Datenquellen.

Öffnen

QueryProvider

Daten laden.

Öffnen

BusinessObjectProvider

BOs schreiben.

Öffnen