DatasourceProvider

Language: Deutsch · **English**

DatasourceProvider

> **Run data sources from rules – run(), DatasourceResult, rows/columns and database functions.**

REFERENZ | DEVELOPER | UPDATED: JUL 2026 | APPLIES TO NUCLOS 4.2026.X

On this page

What is the DatasourceProvider for?

The `DatasourceProvider` executes a **data source** (from Report and form) programmatically.

run(datasourceClass[, params])

Runs the data source and returns a `DatasourceResult` with rows (`rows`) and columns (`columns`). Parameters are optionally passed as a `Map`.

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

> **Oracle**
>
> On Oracle this only works for data sources that **do not write data** – a SELECT there must not modify data.

Evaluating the result

`rows` is a list of `Object[]`; via `getColumns()` you get name and data type per column and can cast accordingly.

```
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());
            }
        }
}
```

Calling a database function

The provider replaces the old `callDBFunction`: the function is wrapped in a data source and called indirectly.

```
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;
    }
}
```

Related pages

Data sources

Data sources.

Open

QueryProvider

Daten laden.

Open

BusinessObjectProvider

BOs schreiben.

Open