Holon Platform JPA QueryDSL Module - Reference manual

Table of Contents

1. Introduction

This module provides a set of QueryDSL integration features for the JPA Datastore module.

The required QueryDSL version is **4.x** or higher.

You must provide the QueryDSL artifacts dependencies in classpath, since they are not provided by this module.

Maven coordinates:

```
<groupId>com.holon-platform.jpa</groupId>
<artifactId>holon-datastore-jpa-querydsl</artifactId>
<version>5.5.0</version>
```

1.1. Sources and contributions

The Holon Platform **JPA QueryDSL integration** module source code is available from the GitHub repository https://github.com/holon-platform/holon-datastore-jpa-querydsl.

See the repository `README` file for information about:

- The source code structure.
- How to build the module artifacts from sources.
- Where to find the code examples.
- How to contribute to the module development.

2. Obtaining the artifacts

The Holon Platform uses Maven for projects build and configuration. All the platform artifacts are published in the **Maven Central Repository**, so there is no need to explicitly declare additional repositories in your project `pom` file.

A **BOM (Bill Of Materials)** `pom` is provided to import the available dependencies for a specific version in your projects.

The BOM can be imported in a Maven project in the following way:

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.holon-platform.jpa</groupId>
      <artifactId>holon-datastore-jpa-querydsl-bom</artifactId>
      <version>5.5.0</version>
      <strong><type>pom</type></strong>
      <strong><scope>import</scope></strong>
    </dependency>
  </dependencies>
</dependencyManagement>
```

2.1. Using the Platform BOM

The Holon Platform provides an **overall Maven BOM (Bill of Materials)** to easily obtain all the available platform artifacts.

See Obtain the platform artifacts for details.

3. Data targets

The QueryDslTarget interface can be used to create a DataTarget using a QueryDSL `EntityPath`,

which represents a JPA *entity* root path.

```
final QueryDslTarget<Test> TARGET = QueryDslTarget.of(QTest.test); ①

Datastore datastore = getDatastore(); // build or obtain a JPA Datastore

datastore.refresh(TARGET, value); ②
```

① Create a `DataTarget` using the QueryDSL `EntityPath` for the JPA `Test` *entity* class

② Use the target with a `Datastore`

4. Properties

The QueryDslProperty interface can be used to create a `PathProperty` from a QueryDSL `Path`. The `of(Path<T> path)` method returns a builder which can be used to setup property configuration and attributes and automatically set the root `EntityPath` as parent property path.

```
final QueryDslProperty<Long> ID = QueryDslProperty.of(QTest.test.id); ①
final QueryDslProperty<String> NAME = QueryDslProperty.of(QTest.test.name); ②

Datastore datastore = getDatastore(); // build or obtain a JPA Datastore

datastore.save(QueryDslTarget.of(QTest.test),
    PropertyBox.builder(ID, NAME).set(ID, 1L).set(NAME, "TestName").build()); ③
```

① Create a `PathProperty` using the QueryDSL `Path` of the JPA `Test` *entity* class `id` attribute of type `Long`

② Create a `PathProperty` using the QueryDSL `Path` of the JPA `Test` *entity* class `name` attribute of type `String`

③ Use the created properties within a `PropertyBox` in a Datastore operation

5. QueryDSL JPA Datastore Commodity

The QueryDSL JPA integration module automatically registers the QueryDsl JPA Datastore **commodity**, which provides a set of methods to build QueryDSL **queries** and **bulk** operation executors.

To obtain the *commodity* the standard Datastore `create(…)` method can be used:

```
Datastore datastore = getDatastore(); // build or obtain a JPA Datastore

QueryDsl queryDslCommodity = datastore.create(QueryDsl.class); ①

JpaQuery<?> query = queryDslCommodity.query(); ②
query = queryDslCommodity.selectFrom(QTest.test); ③

queryDslCommodity.update(QTest.test).set(QTest.test.name, "UpdatedName").where(QTest
.test.id.eq(1L)).execute(); ④
queryDslCommodity.delete(QTest.test).where(QTest.test.id.loe(1L)).execute(); ⑤
```

① Create a `QueryDsl` commodity

② Obtain a query

③ Obtain a query setting the from clause

④ Configure and execute a bulk update

⑤ Configure and execute a bulk delete

The JpaQuery class is an extension of the default QueryDSL JPA query and allows to mix QueryDSL expressions and predicates with standard platform query expressions, such as `QueryFilter`, `QuerySort` and `QueryAggregation`.

```
Datastore datastore = getDatastore(); // build or obtain a JPA Datastore

final PathProperty<Long> ID = QueryDslProperty.of(QTest.test.id); ①
final StringProperty NAME = StringProperty.create("name"); ②

long count = getDatastore().create(QueryDsl.class).query().from(QTest.test)
    .filter(ID.gt(2L).and(NAME.startsWith("n"))).count(); ③
```

① Create a `PathProperty` using a QueryDSL `Path`

② Create a `PathProperty` using standard `create` method, providing property name and type

③ Create and execute a QueryDSL `JpaQuery`, mixing QueryDSL expressions and standard `PathProperty`s

6. Spring Boot starters

The following *starter* artifacts are available to provide a quick project configuration setup using Maven dependency system:

1. The **QueryDSL JPA starter using Hibernate** provides dependencies to the default Holon JPA Datastore Spring Boot starter using the **Hibernate ORM** as persistence provider, in addition to the Holon Platform QueryDSL integration dependencies.

Maven coordinates:

```
<groupId>com.holon-platform.jpa</groupId>
<artifactId>holon-starter-jpa-querydsl-hibernate</artifactId>
<version>5.5.0</version>
```

2. The **QueryDSL JPA starter using Eclipselink** provides dependencies to the default Holon JPA Datastore Spring Boot starter using the **Eclipselink** as persistence provider, in addition to the Holon Platform QueryDSL integration dependencies.

Maven coordinates:

```
<groupId>com.holon-platform.jpa</groupId>
<artifactId>holon-starter-jpa-querydsl-eclipselink</artifactId>
<version>5.5.0</version>
```

7. Loggers

By default, the Holon platform uses the SLF4J API for logging. The use of SLF4J is optional: it is enabled when the presence of SLF4J is detected in the classpath. Otherwise, logging will fall back to JUL (`java.util.logging`).

The logger name for the **JPA Datastore QueryDSL integration** module is `com.holonplatform.datastore.jpa`.

8. System requirements

8.1. Java

The Holon Platform JPA Datastore QueryDSL module requires Java **8** or higher.

8.2. QueryDSL

QueryDSL JPA version **4.x** is required and must be available in classpath to use the JPA Datastore QueryDSL module.