DataNucleus Object Store

Table of Contents

Chapter 1. DataNucleus Object Store

The DataNucleus Object Store enables domain objects to be persisted to relational as well as NoSQL databases. The object store is implemented using DataNucleus.

This user guide discuss end-user features, configuration and customization of the DataNucleus object store.

DataNucleus as a product also supports the JPA API; Apache Isis is likely to also support JPA in the future.

Unresolved directive in ugodn.adoc - include::_ugbtb_datanucleus_overriding-jdo-annotations.adoc[leveloffset=+1] Unresolved directive in ugodn.adoc - include::_ugbtb_datanucleus_java8.adoc[leveloffset=+1]

1.1. Other Guides

Apache Isis documentation is broken out into a number of user and reference guides.

The user guides available are:

- Fundamentals
- Wicket viewer
- Restful Objects viewer
- DataNucleus object store (this guide)
- Security
- Testing
- Beyond the Basics

The reference guides are:

- Annotations
- Domain Services
- Configuration Properties
- Classes, Methods and Schema
- Apache Isis Maven plugin
- Framework Internal Services

The remaining guides are:

- Developers' Guide (how to set up a development environment for Apache Isis and contribute back to the project)
- Committers' Guide (release procedures and related practices)

Chapter 2. Overriding JDO Annotations

The JDO Objectstore (or rather, the underlying DataNucleus implementation) builds its own persistence metamodel by reading both annotations on the class and also by searching for metadata in XML files. The metadata in the XML files takes precedence over the annotations, and so can be used to override metadata that is "hard-coded" in annotations.

For example, as of 1.9.0 the various Isis addons modules (not ASF) use schemas for each entity. For example, the `AuditEntry` entity in the audit module is annotated as:

```
@javax.jdo.annotations.PersistenceCapable(
        identityType=IdentityType.DATASTORE,
        schema = "IsisAddonsAudit",
        table="AuditEntry")
public class AuditEntry {
    ...
}
```

This will map the `AuditEntry` class to a table `"IsisAddonsAudit"."AuditEntry"`; that is using a custom schema to own the object.

Suppose though that for whatever reason we didn't want to use a custom schema but would rather use the default. We can override the above annotation using a `package.jdo` file, for example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<jdo xmlns="http://xmlns.jcp.org/xml/ns/jdo/jdo"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/jdo/jdo
        http://xmlns.jcp.org/xml/ns/jdo/jdo_3_0.xsd" version="3.0">
    <package name="org.isisaddons.module.audit.dom">
        <class name="AuditEntry" schema="PUBLIC" table="IsisAddonsAuditEntry">
        </class>
    </package>
</jdo>
```

This file should be placed can be placed in `src/main/java/META-INF` within your application's `dom` module.

You can use a mixin action on `Persistable` mixin to download the JDO class metadata in XML form.

- The same approach should work for any other JDO metadata, but some experimentation might be required.+

  For example, in writing up the above example we found that writing `schema=""` (in an attempt to say, "use the default schema for this table") actually caused the original annotation value to be used instead.

- Forcing the schema to "PUBLIC" (as in the above example) works, but it isn't ideal because the name "PUBLIC" is not vendor-neutral (it works for HSQLDB, but MS SQL Server uses "dbo" as its default).
- As of 1.9.0 Apache Isis will automatically (attempt) to create the owning schema for a given table if it does not exist. This behaviour can be customized, as described in the section on using modules.
- You may need to override the entire class metadata rather than individual elements; the mixin mentioned above can help here.

Chapter 3. Java8

DataNucleus 4.x supports Java 7, but can also be used with Java 8, eg for streams support against collections managed by DataNucleus.

Just include within `<dependencies>` of your `dom` module's `pom.xml`:

```
<dependency>
    <groupId>org.datanucleus</groupId>
    <artifactId>datanucleus-java8</artifactId>
    <version>4.2.0-release</version>t
</dependency>
```

The DataNucleus website includes a page listing version compatibility of these extensions vis-a-vis the core DataNucleus platform.