Add A JDBC Driver

This document describes the process to add a new JDBC data base driver to DigDash Enterprise.

Requirement: The JDBC driver as a JAR file, and its documentation

I. INSTALLATION

1/ (optional, if the webapps have not been already deployed). Start your DigDash Enterprise Server and wait for the webapps to be fully deployed

2/ Stop the server

3/ copy the driver's JAR file into the following folder:

<DD Install>\apache-tomcat\webapps\ddenterpriseapi\WEB-INF\lib

4/ The JDBC driver must be registered within DigDash Enterprise Server. To do this, edit the following file:

<DD Install>\apache-tomcat\webapps\ddenterpriseapi\WEB-INF\classes\resources\config\sqldriverrepository.xml

Add an XML entry to the sqldriverrepository.xml that looks like the following sample:

```
<SQLDriver    id="MY_DRIVER"
              name="My Driver"
              url="mydriver:"
              manufacturer="My Driver Company"
              class="com.mydriver.MyDriver"
              urlsample="jdbc:mydriver:<database>?<options>"
              availability="both">
        <properties></properties>
</SQLDriver>
```

*Important: Some characters are reserved in XML: '**&**', '**"**', '**<**' or '**>**'*
So if you use some of these characters, make sure you encode them to their corresponding XML entities:

- **&** => **&**
- **"** => **"**
- **<** => **<**
- **>** => **>**

  Example:
  - (WRONG) urlsample="jdbc:mydriver:dbtest?opt1=0&opt2="value""
  - (RIGHT) urlsample="jdbc:mydriver:dbtest?opt1=0&opt2="value""

Parameters:

- **id**: An identifier used by DigDash internally, choose any non existent string, the convention is uppercase, no spaces

- **name**: The driver's name to be displayed in the user interface of the administration console

- **url**: jdbc URL prefix (without "jdbc:"). See driver's documentation

- **manufacturer**: Name of the driver's vendor/developer

- **class**: java class path of the main driver class. See driver's documentation. Optional: JDBC drivers compliant with JDBC 4 do not need a driver class. Keep the class attribute empty (class="") in that case.

- **urlsample**: User friendly URL sample for the user interface of the administration console.

- **availability**: reserved. Leave it at "both".

Important: The deployment is lost when upgrading ddenterpriseapi.war file.
Please follow the deployment guide each time you upgrade DigDash Enterprise

You can find some sample of XML for drivers not provided in DigDash enterprise in **sqldriverrepository.xml** file.

Specific properties (advanced)

You can specify secific properties on some JDBC drivers. These properties are written inside the <properties></properties> tag, under the following format:

<properties>
  <property name="property_name" value="property_value" />
<properties>

Supported properties are:

FORCE_FORWARD_ONLY (undefined | false | true)

Description: specifies the type of JDBC result set cursor used by the Studio for the preview of the SQL query results. By default the Studio uses a TYPE_INSENTIVE_SCROLL cursor to preview the results, but some of the databases do not support this type of cursor. If your driver or database only supports TYPE_FORWARD_ONLY cursors you can specify it with FORCE_FORWARD_ONLY property. Possible values:

- **false** (or **undefined property**): The type of cursor is automatic, TYPE_SCROLL_INSENTIVE in most of the cases, except for HIVE, IMPALA and SAPHANA
- **true**: The type of cursor used in the Studio is TYPE_FORWARD_ONLY

PING_SQL (undefined | SQL query | empty string)

Description: DigDash Enterprise tests the connection with the database by using the JDBC method **Connection.isValid()**. On some drivers this method does not work. In that case DigDash uses an alternative method to “ping” the database, usually by sending a “select 1” statement.
The PING_SQL property allows you to specify this SQL statement depending on your driver or database. Possible values:

- **Undefined property**: The alternative ping query is automatically decided by DigDash: “select 1” except for ORACLE, FIREBIRD, SAPHANA, DB2_AS400 or DB2 drivers
- **SQL query (not empty)**: the specified query is used to ping the databse. Example:
  <property name="PING_SQL" value="select 1 from all_tables" />
- **Empty string**: Special case used to deactivate the ping in the case the JDBC method Connection.isValid() fails. The database is considered to be always accessible. Example:
  <property name="PING_SQL" value="" />

USE_FETCH_FIRST_IN_STUDIO (undefined | false | true)

Description: This property is used only for preview the SQL result in the data source configuration dialog (Studio). It modifies the query by adding “FETCH FIRST n ROWS ONLY” to it (n is replaced byt the number of preview rows). It is useful for drivers that do not support JDBC’s **Statement.setMaxRows(n)**, for example AS400 JDBC driver. Possible values:

- **false** (or **undefined property**): The preview limit is defined by using JDBC method **Statement.setMaxRows(n)**
- **true**: Preview limit is defined by adding FERTCH FIRST n ROWS ONLY to the SQL in the Studio.

FORBID_POOL_CONNECTION (undefined | false | true)

Description: Prevents DigDash Enterprise from using a JDBC connection pool for this driver. A JDBC connection pool optimizes access to the data base by keeping connections in an opened state, and reuse them for different queries. In some cases it is preferred to not use a JDBC connection pool, for example to ensure that connections will not stay opened too long on the data base after they have been used for a query. This property answers that need. Possible values:

- **false** (or **undefined property**): A JDBC connection pool may be used by the driver
- **true**: The JDBC connection pool will not be used by the driver and each SQL query will have its own connection.

