Using DODS

Table of Contents

Chapter 1. Data Object Design Studio

The Data Object Design Studio (DODS), shown in Figure 1, is a tool which, for the given doml file, can generate SQL script files for creating tables (for each table separately and one cumulative file for creating all tables), one file for deleting all tables, and/or java code for data objects described in the given doml file. DODS also has possibility to compile generated java classes and to parse SQL files (to split cumulative SQL into more separated SQL files using SQLSplitter tool).

Figure 1: DODS Generator Wizard

Data objects described in the given DOML file correspond to tables in the database. Each data object has attributes, which describe database columns, and reference attributes, which refer to other data objects. Reference attributes let you create a hierarchy of data objects (for example, many-to-one or many-to-many relationships).

For the given DOML, DODS generates all of the code to implement it. For example:

- SQL code to define the database tables

- Java code to create the corresponding application data objects

For each data object, DODS generates a set of source files. For example, if your DOML file includes the definition of an entity named "thing," then DODS would generate the following:

- A file named thing.sql containing the SQL CREATE TABLE command to construct a table in a relational database.
- Java source file defining a data object representing a row in the table.

  This class provides a "set" and "get" method for each attribute, methods to handle caching, and is a subclass of the Enhydra framework class GenericDO. In this example, the class would be named ThingDO.
- Java source file that defines a query class, which provides SQL query access to the database table.

  The query class returns a collection of ThingDO objects that represent the rows found in the table matching criteria passed from the application..

DODS is one part of Enhydra 5.1. If Enhydra 5.1 is installed, so is DODS. In this case, DODS home directory <dods_home> is: <enhydra_home>/dods.

Since this version, DODS has become independent from Enhydra, which means that can be used without it. In this case, DODS home directory <dods_home> is the directory in which independent DODS is installed.

DODS independence is detailly explained in the chapter DODS independence.

Chapter 2. DOML file syntax

This chapter describes the syntax of DOML files, which are used by the Data Object Design Studio (DODS) to generate data access code for Enhydra applications.

Structure

The hierarchy of tags in a DOML file is:

```
<doml>
  <database>
    <package>
      <package>
        ....
      </package>
      <table>
        <column>
          <type/>
          <referenceObject/>
          <initialValue/>
          <javadoc/>
        </column>
        <index>
          <indexColumn/>
        </index>
      </table>
    </package>
  </database>
</doml>
```

The references of these tags are described in "DOML tag reference" (html [using_dods/doml_tags.html] , pdf [using_dods/doml_tags.pdf])

Sample DOML file

The following snippet shows content of a DOML file, sample.doml, which creates tables containing data about cars, car dealers, and car owners.

```
<?xml version="1.0"encoding="UTF-8"?>
<doml>
  <database database="Standard"templateset="standard">
    <package id="sample">
```

```
<table id="sample.Dealer">
  <column id="Name">
    <type dbType="VARCHAR"javaType="String"/>
  </column>
</table>
<table id="sample.Owner">
  <column id="Name">
    <type dbType="VARCHAR"javaType="String"/>
  </column>
  <column id="Age">
    <type dbType="INTEGER"javaType="int"/>
  </column>
</table>
<table id="sample.Car">
  <column id="LicensePlate">
    <type dbType="CHAR"javaType="String"/>
  </column>
  <column id="Dealer">
    <referenceObject reference="sample.Dealer"/>
    <type dbType="none"javaType="sample.DealerDO"/>
  </column>
</table>
<table id="sample.CarOwner">
  <column id="Car">
    <referenceObject reference="sample.Car"/>
    <type dbType="none"javaType="sample.CarDO"/>
    </column>
  <column id="Owner">
    <referenceObject reference="sample.Owner"/>
    <type dbType="none"javaType="sample.OwnerDO"/>
   </column>
  <column id="IsCurrent">
    <type dbType="BIT"javaType="boolean"/>
  </column>
  <index id="index_1" unique="true">
    <indexColumn id="Car"/>
```

```
        </index>
      </table>
    </package>
  </database>
</doml>
```

Chapter 3. Starting dods generator

There are two different ways to run dods generator. If you want to start generator quickly, you can start wizard by typing

```
dods (for Windows)
```
or

```
./dods (for Linux)
```
without any parameter. Those files are located in

- <enhydra_home>/bin folder, for DODS in Enhydra.
- <dods_home>/bin folder, for independent DODS

- Note:

  <enhydra_home>/bin (in the case DODS is used in Enhydra), or <dods_home>/bin folder (for independent DODS) should be added in the system path. Then, DODS can be started from any directory (by typing dods).

This will be described in the second chapter "Quick Compile" (html [using_dods/dods_start.html#ch2], pdf [using_dods/dods_start.pdf]) of the document "Starting DODS" (html [using_dods/dods_start.html] , pdf [using_dods/dods_start.pdf]).

If you want to start generator without wizard, you need to type (in the command line) dods with additional parameters. You can find details in the third chapter "Custom Compile" (html [using_dods/dods_start.html#ch3], pdf [using_dods/dods_start.pdf]) of the document "Starting DODS" (html [using_dods/dods_start.html] , pdf [using_dods/dods_start.pdf]).

For more details see document "Starting DODS" (html [using_dods/dods_start.html] , pdf [using_dods/dods_start.pdf]) mentioned previous in this chapter.

Chapter 4. DODS independence

Since this version, DODS is independent from Enhydra. This means that it is possible for user to make any application (it doesn't need to be enhydra application) that can use DODS.

DODS works with DatabaseManagers. DatabaseManager is class that provides facilities for work with databases.

There are two modes of using DODS:

- non-threading

  In non-threading mode, only one DatabaseManager is used for the whole application, no matter the application has one or more Threads.

- threading

  In threading mode, there is one DatabaseManager for every Thread. User needs, for every Thread, to define DatabaseManager. If, for any Thread, the DatabaseManager is not defined, the default DatabaseManager is used.

In the following text, the DODS independence is explaned for non-threading mode.

To make non-enhydra application that can use DODS, the following things must be done:

- in main application,

  add code that makes new DatabaseManager and registers it in DODS:

```
try {
    . . .
    String fileName = "discRack.conf";
    DatabaseManager dbManager =
        StandardDatabaseManager.newInstance(fileName);
    DODS.register(dbManager);
    . . .
} catch (Exception e) {
    e.printStackTrace();
}
```

  where "discRack.conf" is an example of application's configuration file. This file is the same as the Database Manager section of Enhydra application's configuration file.

This file can look like this:

```
#
# The databases that are used by CSAM.  Each of these databases
# has configuration parameters set under DatabaseManager.DB."databaseName".
#
```

```
DatabaseManager.Databases[] = "sid1"
#
# The default database used in this application.
#
DatabaseManager.DefaultDatabase = "sid1"
#
# Turn on/off debugging for transactions or queries. Valid values
# are "true" or "false".
#
DatabaseManager.Debug = "false"
#
# The type of database. Normally this is "Standard".
#
DatabaseManager.DB.sid1.ClassType = "Standard"
# DatabaseManager.DB.sid1.ClassType = "Oracle"

#
# The jdbc driver to use.
#
DatabaseManager.DB.sid1.JdbcDriver = "org.enhydra.instantdb.jdbc.idbDriver"
# DatabaseManager.DB.sid1.JdbcDriver = "oracle.jdbc.driver.OracleDriver"
# DatabaseManager.DB.sid1.JdbcDriver = "sun.jdbc.odbc.JdbcOdbcDriver"

#
# Database url.
#
DatabaseManager.DB.sid1.Connection.Url = "jdbc:idb:@OUTPUT@/discRack.prp"
# DatabaseManager.DB.sid1.Connection.Url = "jdbc:oracle:thin:@MyHost:MyPort:MyDBName"
# DatabaseManager.DB.sid1.Connection.Url = "jdbc:odbc:discRack"

#
# Database user name.  All connection are allocated by this user.
#
DatabaseManager.DB.sid1.Connection.User = "scott"
#DatabaseManager.DB.sid1.Connection.User = "Admin"

# Database user password.
#
```

```
DatabaseManager.DB.sid1.Connection.Password = "tiger"
#DatabaseManager.DB.sid1.Connection.Password = ""

#
# The maximum number of connections that a connection
# pool will hold.  If set to zero, then connections
# are allocated indefinitly or until the database
# refuses to allocate any new connections.
#
DatabaseManager.DB.sid1.Connection.MaxPoolSize = 30

#
# Maximum amount of time that a thread will wait for
# a connection from the connection pool before an
# exception is thrown.  This will prevent possible dead
# locks.  The time out is in milliseconds.  If the
# time out is <= zero, the allocation of connections
# will wait indefinitely.
#
DatabaseManager.DB.sid1.Connection.AllocationTimeout = 10000

#
# Used to log database (SQL) activity.
#
DatabaseManager.DB.sid1.Connection.Logging = false

#
# The number of object identifiers that are allocated
# as a group and held in memory.  These identifiers
# are assigned to new data objects that are inserted
# into the database.
#
DatabaseManager.DB.sid1.ObjectId.CacheSize = 20
DatabaseManager.DB.sid1.ObjectId.MinValue = 1000000

#
# User wildcards
#
```

```
DatabaseManager.DB.User.userWildcard = "*"
DatabaseManager.DB.User.userSingleWildcard = "_"
DatabaseManager.DB.User.userSingleWildcardEscape = "$"
DatabaseManager.DB.User.userWildcardEscape = "$"

#
# Cache configuration
#
DatabaseManager.DB.Cache.defaultMaxCacheSize = 100
DatabaseManager.DB.Cache.maxCacheSize.Disc = 30
DatabaseManager.DB.Cache.maxCacheSize.Person = 10
```

The example of non-enhydra application that can use DODS is DiscRack application, explained in next section.

Examples of non-enhydra applications

Examples of DODS non-enhydra applications are included in DODS installation and they are in DODS, in directory:

```
<DODS_HOME>/examples
```

Process of running non-enhydra application will be presented in this section on the example Disc Rack. This example application is in <DODS_HOME>/examples/discrack directory.

To run example , these steps have to be done in Command Promt:

- first, go to wanted example (directory)

  ```
  cd <DODS_HOME>/examples /discrack
  ```

- second, run ant,by typing:

  ```
  ant
  ```
  ant will build this application in its <output_directory>

- then, go to application's output directory:

  ```
  cd <output_directory>
  ```

- then, example will be run with:

  ```
  run
  ```

The ant, which is used here, must be DODS's ant.bat, which means that path <DODS_HOME>/bin must be included at the beginning of the system path.

Chapter 5. Caching

Caching affects the behaviour of the DO class. If checked, all DO instances are stored in the cache inside the DO class. Subsequent queries of the table use the Query class for queries. The results of all Queries, no matter which type of caching is used, are complete.

When you insert new DO into the database, the DO is also (automatically) inserted into the cache.

When you delete DO from the database, the DO is also (automatically) deleted from the cache.

The possible values for caching are:

- None

  This flag means that there is no caching available.

- Partial

  This flag means that, when you run Query, next actions are performed:

  - First, it is checked if this select (Query) had been ran before. If yes, the results are retrieved by searching only the cache. If not, the results are retrieved by searching only the database.
  - There is one special case - Query by OID. In this case also, first is checked whether this Query had been ran before. If yes, the results are retrieved by searching only the cache. If not, first is performed searching the cache, and if the result (there is only one DO that can be result of Query by OID because only one DO can have specified OID) is not found, hitting the database is performed. It is done this way because there is a possibility that any non-OID Query has put the DO (DO with wanted OID) in the cache as one of its results.
  - The results (data objects) from the database (if hitting the database was performed) are then being added in the cache (added to the DOs that are already in the cache). This means that the size of the cache, with every query that hits the database and gets results, is grooving.

- LRU

  This flag means that, when you run Query, next actions are performed:

  - If Query by OID is made, DOs are being searched in the cache. If the DO that matches the Query is not found in the cache, hitting the database is performed, but if the DO is found in the cache, it is returned as a Query result.
  - If non-OID Query is made, DOs are only being searched in the database, and DOs from database that match the Query are returned as the Query result.
  - The results (data objects) from the database (if hitting the database was performed) are then being added in the cache. The size of the cache is limited by the maximal number of data objects that can be stored in it. When the cache is full, the DOs in it are being replaced by new data objects according to LRU (least recently used) algorithm. This algorithm says that the DO (data object) which had been used the least recently (in the scale of time, the DO to which had been accessed the longest time ago) is replaced with the new data object.

- Full

  If this flag is checked, the entire table is queried and cached when your application starts. This is appropriate for

tables of "static" data which is accessed frequently and which will not change during the execution of your application. In this case, all Queries search the cache, hitting the database is never performed (because all database DOs are stored in the cache).

Default value of caching is "none".

Cache size

The size of the cache is defined in application's configuration file.

Example:

For file discRack.conf with type of caching LRU, part of code for cache size can look like:

```
#
# Cache configuration
#
DatabaseManager.DB.Cache.defaultMaxCacheSize = 100
DatabaseManager.DB.Cache.maxCacheSize.Disc = 30
DatabaseManager.DB.Cache.maxCacheSize.Person = 10
```

In the following text are explained possible sizes for every type of cache.

- Partial

  The parameter maxCacheSize of application's configuration file defines size of the cache.

  - maxCacheSize > 0

    This cache is limited. The maximal number of DOs is maxCacheSize.

  - maxCacheSize = 0

    This means that there is no cache available. This value excludes cache from use.

  - maxCacheSize < 0

    This cache is unlimited.

  If parameter maxCacheSize is not defined in configuration file, the size of the cache is defined by parameter defaultMaxCacheSize of the same configuration file. The rules for this value are the same as for maxCacheSize explained before.

  If neither parameter maxCacheSize nor parameter defaultMaxCacheSize is defined, DODS takes by its default that cache is unlimited.

  As explained, the results from the database are added to DOs that are already in the cache, so the size of the cache, with every query that hits the database and gets results, is grooving.

In unlimited cache, in order not to make this cache too large, when half of memory resource is reached, the cache is being refreshed (emptied) by using the method refreshCache().

Of course, if cache is limited, it is being refreshed when maximal number of DOs in the cache is reached by using the same method refreshCache().

- LRU

  The parameter maxCacheSize of application's configuration file defines size of the cache.

  - maxCacheSize > 0

    This cache is limited. The maximal number of DOs is maxCacheSize.

  - maxCacheSize = 0

    This means that there is no cache available. This value excludes cache from use.

  - maxCacheSize < 0

    This cache is unlimited.

  If parameter maxCacheSize is not defined in configuration file, the size of the cache is defined by parameter defaultMaxCacheSize of the same configuration file. The rules for this value are the same as for maxCacheSize explained before.

  If neither parameter maxCacheSize nor parameter defaultMaxCacheSize are defined, DODS has its own constant DEFAULT_MAX_CACHE_SIZE that defines maximal size of the cache.

  When the cache is full, the DOs in it are being replaced by new data objects according to LRU algorithm.

- Full

  If paramether maxCacheSize is 0, or this parameter in not defined in configuration file and parameter defaultMaxCacheSize is 0, the cache is excluded.

  In any other way, the cache is unlimited.

Chapter 6. User wildcards

Like cache size, user wildcards are also defined in application's configuration file.

Example:

For file discRack.conf part of code for user wildcards can look like:

```
#
# User wildcards
#
DatabaseManager.DB.User.userWildcard = "*"
DatabaseManager.DB.User.userSingleWildcard = "_"
DatabaseManager.DB.User.userSingleWildcardEscape = "$"
DatabaseManager.DB.User.userWildcardEscape = "$"
```

Chapter 7. Database Independency

DODS generates java code that is database independent. This means that java code is the same no matter which base you use.

When you want to change the database, the only thing you need to do is to change <App_name>.conf file (update it with information considering new database). This change is necessary for connection to database.

Chapter 8. Using multi databases in DODS

Enhydra has the possibility of working with more than one database at the same time. This means that, when the application is started, you don't have to stop it in order to change the database the application uses.

If you want to use this Enhydra option, you must (in <App_name>.conf file) define all logical databases you want to use. For each of these databases you must configurate all needed parameters. If you don't want to use this Enhydra option, the default database will be used.

When you update <App_name>.conf (with information about all databases) and start you application, it uses the default database. The definition of the new (desired) database is being done in the stage of creation of DO and Query objects.

When you create Query object for a database (given or default), the result of this Query are only DOs from that database, not from any other base.

If caching is used, there is only one cache for all <object_class>DOs (<object_class>DOs from all databases are placed in the same cache).

When you create DO for a specific database, this DO can't change database any more. If you want to translate one DO object from its database to another, you must create new DO in that another database, and then copy data of DO you want to translate into new DO (there are copy methods which you can use for this). In this way, new DO gets its own ID of its base.

Query object can change database. When you change database of the Query object, now the result of this Query will be DOs from this new database; you won't be able to get the DOs from the previous database any more.

DODS takes care of referential integrities within the database which means that DODS searches referenced object in the same database in which the object that referenced it is. If you want to use referenced objects from any other database, you must yourself take care of referential integrities.

In the <object_class>DO class public constructors and methods (loadData, createVirgin, createForExisting, createExisting) are now defined and with the parameter database. You can use them with this parameter in which case the object will be created for the given logical database, or you can use these constructors and methods without database parameter. In this case, they will be created for default database.

- Tip: Be very careful when you add DO objects in the databases, or when you use Query objects. Now, there is more than one database that is used, and it's more difficult to track in which database, which DO object is placed, and for which base you run Query.

- Tip: When you use multi databases with full caching, it would be better to announce in advance which databases will be used (with the method useLogicalDatabase(String database)) so that all needed DOs would be put in the cache at once, not partial.

Chapter 9. Conversion of doml file

DODS has the possibility of converting doml file, release 5.*. As described in the second chapter "Quick Compile" (html [using_dods/dods_start.html#ch2], pdf [using_dods/dods_start.pdf]) of the document "Starting DODS" (html [using_dods/dods_start.html] , pdf [using_dods/dods_start.pdf]), Generator Wizard has a possibility of converting doml file into four document types: html, pdf, xmi and ptl. The name of the target (html, pdf, xmi, ptl) files will be the same as the name of doml file that is being converted, and they would be located in output directory.

The doml file can also be converted manually. For this purpose are used files in <dods_root>/bin folder, and they are:

- doml2html - converts doml 5.* file into html file.

  doml2html is used with the following parameters:

  ```
  doml2html [-help] [doml5*-file] [html-file]
  ```

  where:

  - help - prints message for usage and exits.
  - doml5*-file - doml file relese 5.*.
  - html-file - desired target html file.
- doml2pdf - converts doml 5.* file into pdf file.

  doml2pdf is used with the following parameters:

  ```
  doml2pdf [-help] [doml5*-file] [pdf-file]
  ```

  where:

  - help - prints message for usage and exits.
  - doml5*-file - doml file relese 5.*.
  - pdf-file - desired target pdf file.
- doml2xmi - converts doml 5.* file into xmi file.

  doml2xmi is used with the following parameters:

  ```
  doml2xmi [-help] [doml5*-file] [xmi-file]
  ```

  where:

  - help - prints message for usage and exits.
  - doml5*-file - doml file relese 5.*.
  - xmi-file - desired target xmi file.
- doml2ptl - converts doml 5.* file into ptl file.

doml2ptl is used with the following parameters:

```
doml2ptl [-help] [doml5*-file] [ptl-file]
```

where:

- help - prints message for usage and exits.
- doml5*-file - doml file relese 5.*.
- ptl-file - desired target ptl (Rational Rose) file.

- doml31_2_doml51 - converting doml 3.1 file into doml 5.1 file.

  doml31_2_doml51 is used with the following parameters:

  ```
  doml31_2_doml51  [-help] [doml31-file] [doml51-file]
  ```

  where:

  - help - prints message for usage and exits.
  - doml31-file - doml file relese 3.1.
  - doml51-file - desired target doml file relese 5.*.

Chapter 10. Template sets

User can make its own template sets. All template sets (standard, multidb, webdocwf, multidb_ webdocwf and users) are placed in one directory. The name and location of this directory can be set (or changed) in dods.properties file (in <dods_home> directory). Within this directory, every template set is placed in its own subdirectory.

Chapter 11. Custom Configuration

To configure DODS, use dodsConf.xml file, located in <dods_root>/build/conf directory.

This file contains the following information:

- location of templates - tag <TemplateDir>

  example:

  ```
  <TemplateDir>C:/DODS/build/template</TemplateDir>
  ```

- for each database vendor, location of its configuration file (in xml format) - tag <Database>

  example:

  ```
  <Database>
    <Vendor name="Standard">C:/DODS/build/conf/StandardConf.xml</Vendor>
    <Vendor name="InstantDB">C:/DODS/build/conf/InstantDBConf.xml</Vendor>
    <Vendor name="Oracle">C:/DODS/build/conf/OracleConf.xml</Vendor>
    <Vendor name="Informix">C:/DODS/build/conf/InformixConf.xml</Vendor>
    <Vendor name="MSQL">C:/DODS/build/conf/MSQLConf.xml</Vendor>
    <Vendor name="Sybase">C:/DODS/build/conf/SybaseConf.xml</Vendor>
    <Vendor name="PostgreSQL">C:/DODS/build/conf/PostgreSQLConf.xml</Vendor>
    <Vendor name="HypersonicSQL">C:/DODS/build/conf/HypersonicSQLConf.xml</Vendor>
    <Vendor name="DB2">C:/DODS/build/conf/DB2Conf.xml</Vendor>
    <Vendor name="QED">C:/DODS/build/conf/QEDConf.xml</Vendor>
    <Vendor name="MySQL">C:/DODS/build/conf/MySQLConf.xml</Vendor>
  </Database>
  ```

Database Vendor's configuration file contains informatio about that database (type of ObjectId, column name for oid and version, information about DeleteCascade, constraints, quotes,comments, characterd for like and wildcard, mapping JDBC types to vendor-specific data types,...).

If tags <ClassPath>, <ClassName> are not mentioned, standard code is generated for that database vendor.

If database is specific, path to jar file for that database vendor is in tag <ClassPath>, and its main class is in tag <ClassName>.

example, for database Informix:

```
<ClassPath>C:/DODS/lib/dbvendors/informix.jar</ClassPath>
```

```
<ClassName>com.lutris.appserver.server.sql.informix.InformixLogicalDatabase</ClassName>
```