Starting DODS

Table of Contents

Chapter 1. File location

After generating, locations of generated files are:

```
<OUTPUT_DIRECTORY>\SQLcreate.sql

<OUTPUT_DIRECTORY>\<PACKAGE_0>\..\<PACKAGE_N>\<TableName>DataStruct.java

<OUTPUT_DIRECTORY>\<PACKAGE_0>\..\<PACKAGE_N>\<TableName>DataStruct.java

<OUTPUT_DIRECTORY>\<PACKAGE_0>\..\<PACKAGE_N>\<TableName>DOI.java

<OUTPUT_DIRECTORY>\<PACKAGE_0>\..\<PACKAGE_N>\<TableName>DO.java

<OUTPUT_DIRECTORY>\<PACKAGE_0>\..\<PACKAGE_N>\<TableName>Query.java

<OUTPUT_DIRECTORY>\<PACKAGE_0>\..\<PACKAGE_N>\<TableName>.xml
```

where <OUTPUT_DIRECTORY> is base directory of your project, <PACKAGE_0>\..\<PACKAGE_N> is generated from last package id attribute of DOML file, and <TableName> is the name of the table from your database. For example, if part of your DOML file looks like this:

```
<package id="discRack">

  lt;package id="discRack.data">

    <table id=" discRack.data.Person" notUsingOId="true" dbTableName="Person">

        ...

    </table>
```

you will get file structure as follows:

```
<OUTPUT_DIRECTORY>\discRack\data\PersonDataStruct.java

<OUTPUT_DIRECTORY>\discRack\data\PersonDOI.java

<OUTPUT_DIRECTORY>\discRack\data\PersonDO.java

<OUTPUT_DIRECTORY>\discRack\data\PersonQuery.java

<OUTPUT_DIRECTORY>\discRack\data\Person.xml
```

There are transient XML files that are generated from DOML file, before Java code is generated. The java code, mentioned before, is actually generated from those transient xml files. If you want, you can change these xml files instead of DOML file and generate Java code directly, without using the DOML file. You can find instructions for this in Advanced Custom Compile section.

If you change the DOML file, all java classes will be generated again, but, if you change transient xml files instead of the DOML file, only changed xml files are generated in java files. Other java files (whose xml files are not changed) are left as they are.

Chapter 2. Quick Compile

DODS Generator Wizard is a graphical tool that helps you to easily generate Java and SQL files. It is recommended for the first time users.

When you start dods.bat you will get window like on Figure 2.

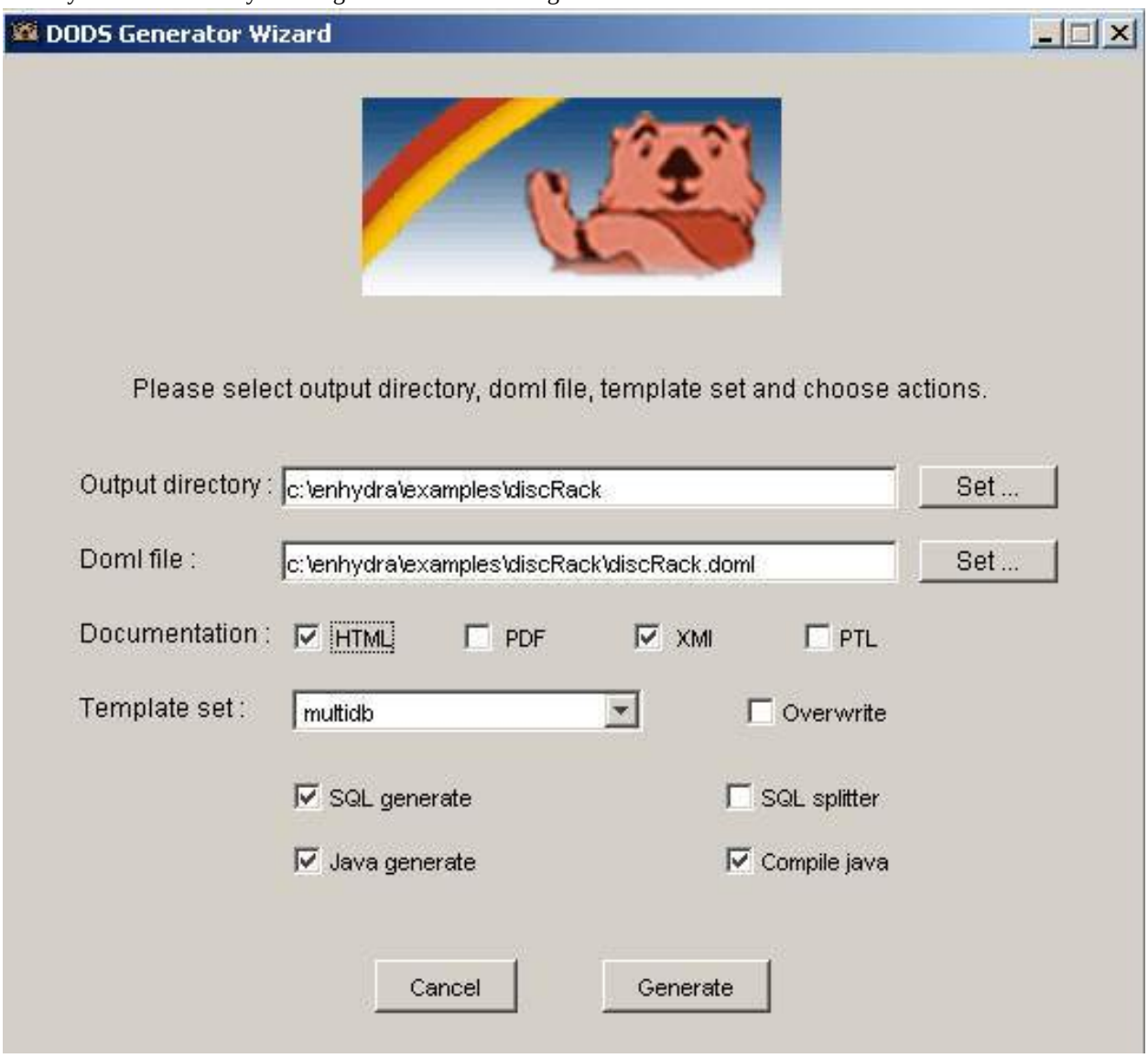

Figure 2: DODS Generator Wizard

In the Output directory field you should input directory with full path of output directory that will be used.

There are four options on the Generator Wizard:

- Generate SQL:

  This field should be checked if you want to generate: SQL files for each table separately, one cumulutave SQL file for creating all tables (SQLcreate.sql), and one file for deleting those tables (SQLdrop.sql).

- SQL Splitter:

  It is used for creating separated cumulative SQL files (for creating tables, for adding foreign keys, primary keys and for deleting tables). This option enables creating tables without cross references, and after their creation, adding needed references.

  SQL Splitter copies all SQL commands from all SQL files which are situated in the working directory and all its subdirectories into SQL files.

  Original SQL files are created by DODS - Enhydra.

  All SQL commands are copied into file separateCreate.sql except sql commands which reference to foreign and primary key columns.

  In the separateIntegrity.sql file class puts ALTER TABLE sql commands with adding foreign key references.

  In the separatePrimary.sql file class puts ALTER TABLE sql commands with adding primary keys.

  In the separateDropTable.sql file class puts DROP TABLE sql commands for all tables which were created by create table SQL statements in the first file (separateCreate.sql).

  In the separateIndex.sql file class puts CREATE INDEX sql commands for all tables which were created by create table SQL statements in the first file(separateCreate.sql).

  In the separateDropIntegrity.sql file class puts DROP foreign key sql commands for all tables which were created by create table SQL statements in the first file (separateCreate.sql).

  In the separateDropPrimary.sql file class puts DROP primary sql commands for all tables which created by create table SQL statements in the first file (separateCreate.sql).

  All others Sql commands class puts into separate file.

  Unless Generate SQL field is checked, this field can not be checked. If this option is checked, Generator Wizard doesn't create cumulative SQL files.

- Generate Java:

  This field should be checked if you want to generate Java files (DO, Query, DOI and DataStruct objects).

  - Compile Java:

    It is used for compiling generated java files. Compiled files will be located in folder /<output_directory>classes. Unless Generate Java field is checked, this field can not be checked.

  If you do not need both Java and SQL generation, you can choose one of them instead of both.

At least one of the Generate fields must be checked.

There is one combo box on the Generator Wizard. It contains following template sets:

- standard:

  If this template set is chosen, DODS generates standard code.

- multidb:

  If this template set is chosen, beside standard code, DODS also generates extra code that provides possibility of working with multi databases. Standard code without this extra code provides possibility of working with one database. You can see description of these new methods that work with multi databases in generated API.

- <user_defined_templates>:

  Users can define their own tempate sets.

Selected template set depends on <template_set> tag in doml file. If this tag is not set, defaut template set is "standard". If this tag is set, the value of this tag will be selected in template set combo box.

There is a possibility on the Generator Wizard for generating four types of documentation:

- HTML:

  If you check this field, doml file will be converted into html file.

- PDF:

  If you check this field, doml file will be converted into pdf file.

- XMI:

  If you check this field, doml file will be converted into xmi file.

- PTL:

  If you check this field, doml file will be converted into ptl (Rational Rose) file.

On the Generator Wizard, there is also a check box:

- overwrite

  for code generating (java and sql), no matter if the code already existed.

Chapter 3. Custom Compile

In case you want to generate Java and SQL code manually, type dods in the command line with desired parameters.

Command line:

```
dods  [-?/help] [-a action] [-t templateset] [-b/-database] [-f/force]
      [-h/html] [-p/pdf] [-x/xmi] [-r/ptl] domlfile outputdir
```

where:

- outputdir is full path to output directory that will be used.
- domlfile is full path to .doml file for generating code.

options:

- [-? -help] shows help.
- [-a action] - ant task parameter for code generation:
  - dods:build_all - to create all sql files and java classes (default).
  - dods:sql - to create only sql files.
  - dods:java - to create only java files and to compile them.
  - dods:javaNoCompile - to create only java files and not to compile them.
  - dods:noCompile - to create SQL files and java files and not to compile them.
  - dods:build_all_split - to create all sql files and java classes and to compile it. SQL files will be divided into separate files using SQLSplitter .
  - dods:sqlsplit - to create only sql files and separate in different files using SQLSplitter.
  - dods:noCompileSplit - to create SQL files and separate sql commands using SQLSplitter and java files and not to compile them.
- [-t templateset] - template set for generating java and sql code:
  - standard - generate standard java code (default).
  - multidb - generate java code with multi database support.
  - <user defined> - any user defined template set.
- [-b/-database] - sets database vendor for generating sql.
- [-f/-force] - with this switch, code will be always generated, without it, only changes will be regenerated.
- [-h/-html] - generates DODS html documentation from .doml file.

- [-p/-pdf] - generates DODS pdf documentation from .doml file.
- [-x/-xmi] - generates DODS xmi documentation from .doml file.
- [-r/-ptl] - generates DODS ptl (Rational Rose) documentation from .doml file.

Chapter 4. Advanced Custom Compile

In this chapter you can find information about advanced settings for generation of Java files.

One XML file is generated for every table from DOML file (situated in table folder with other java and sql files). That XML file is used as a base for generating four Java files.

DTD for that file can be found in <enhydr_home>/dods/dtd/temporaryXML.dtd file. Some tags could be changed, i.e. <AUTHOR>.

- Important: some tags should not be changed, or otherwise generated code will not be compailable.

Structure

The hierarchy of tags in a XML file is, as follows:

```
<TABLE>
  <PACKAGE>
  <AUTHOR>
  <PROJECT_NAME>
  <TABLE_NAME>
  <CLASS_NAME>
  <EXTENDS>
  <DB_VENDOR>
  <TEMPLATE_SET>
  <DO_IS_OID_BASED>
  <IS_ABSTRACT>
  <IS_LAZY_LOADING>
  <DELETE_CASCADES>
  <COLUMN>
    <REFERENCE_OBJECT>
      <CONSTRAINT>
      <IS_ABSTRACT>
      <IS_FOREIGN_KEY>
      <PACKAGE>
    </REFERENCE_OBJECT>
    <IS_CONSTANT>
    <JAVADOC>
    <DB_TYPE>
    <JAVA_TYPE>
```

```
    <JAVA_DEFAULT_VALUE>
    <USED_FOR_QUERY>
    <CAN_BE_NULL>
    <IS_PRIMARY_KEY>
    <IS_FOREIGN_KEY>
    <SIZE>
  </COLUMN>
  <REFERRER>
    <REFATTR/>
  </REFERRER>
  <INDEX>
    <INDEX_COLUMN/>
  </INDEX>
</TABLE>
```

The references of these tags are described in "Tag reference in a XML file" (html [xml_tags.html] , pdf [xml_tags.pdf])

Example

This in en example of how the tags mentioned before can be used. Transient XML can look like this:

```
<TABLE>
  <PACKAGE>discRack/data/disc</PACKAGE>
  <AUTHOR>NN</AUTHOR>
  <PROJECT_NAME>discRack</PROJECT_NAME>
  <TABLE_NAME>Disc</TABLE_NAME>
  <CLASS_NAME>Disc</CLASS_NAME>
  <EXTENDS>com.lutris.dods.builder.generator.dataobject.GenericDO</EXTENDS>
  <DB_VENDOR>Standard</DB_VENDOR>
  <TEMPLATE_SET>standard</TEMPLATE_SET>
  <DO_IS_OID_BASED>true</DO_IS_OID_BASED>
  <IS_ABSTRACT>false</IS_ABSTRACT>
  <IS_LAZY_LOADING>true</IS_LAZY_LOADING>
  <DELETE_CASCADES>false</DELETE_CASCADES>

  <COLUMN name="owner">
    <REFERENCE_OBJECT name="Person">
      <CONSTRAINT>true</CONSTRAINT>
```

```
      <IS_ABSTRACT>false</IS_ABSTRACT>
      <IS_FOREIGN_KEY>false</IS_FOREIGN_KEY>
      <PACKAGE>discRack.data.person</PACKAGE>
    </REFERENCE_OBJECT>
    <IS_CONSTANT>false</IS_CONSTANT>
    <JAVADOC></JAVADOC>
    <DB_TYPE>none</DB_TYPE>
    <JAVA_TYPE>DiscRack.data.person.PersonDO</JAVA_TYPE>
    <JAVA_DEFAULT_VALUE></JAVA_DEFAULT_VALUE>
    <USED_FOR_QUERY>true</USED_FOR_QUERY>
    <CAN_BE_NULL>false</CAN_BE_NULL>
    <IS_PRIMARY_KEY>false</IS_PRIMARY_KEY>
    <IS_FOREIGN_KEY>true</IS_FOREIGN_KEY>
    <SIZE></SIZE>
  </COLUMN>

  <REFERRER name="Storage" package="discRack.data.storage">
    <REFATTR name="disc" do_name="Disc"/>
  </REFERRER>
  <INDEX id="index_owner" unique="true" clustered="false">
    <INDEX_COLUMN id="owner"/>
  </INDEX>
</TABLE>
```

Tags <TABLE> and <EXTENDS> must exist and must NOT be changed.

Tags <AUTHOR>, <PROJECT_NAME> and <DB_VENDOR> must exist and can be changed.

Tags <TEMPLATE_SET>, <JAVADOC> and <JAVA_DEFAULT_VALUE> are not required.

Tags <REFERENCE_OBJECT>, <CONSTRAINT>, <IS_ABSTRACT>, <IS_FOREIGN_KEY>, <PACKAGE>, <REFERRER> and <REFATTR> can exist and must NOT be changed.

Tags <PACKAGE>, <TABLE_NAME>, <CLASS_NAME>, <DO_IS_OID_BASED>, <IS_ABSTRACT>, <IS_LAZY_LOADING>, <DELETE_CASCADES>, <COLUMN>, <IS_CONSTANT>, <DB_TYPE>, <JAVA_TYPE>, <USED_FOR_QUERY>, CAN_BE_NULL>, <IS_PRIMARY_KEY>, <IS_FOREIGN_KEY>, <SIZE>, <INDEX> and <INDEX_COLUMN> must exist and can be changed.