MsSqlDataToText

Export large datasets from MS SQL server to CSV files.

Purpose / Origin

A command line tool to export large MS SQL databases to text files *(*.csv)*. The need for this tool arose - as it's always the case, during a project. Data stored in a MS SQL server database needed to be exported in a format usable by *a Linux shop*.

Non of the out-of-the-box solutions I tried out did work, e.g. Oracle's MySQL Workbench, MS SQL's Export assistant. Neither did various scripts / suggestions found online. The closest solution was a PowerShell script found on StackOverflow.

Due to a) the size of some of the tables and b) column data types such as `binary` and `image`, it failed with a *System.OutOfMemoryException*. But the idea behind it was sound, so I replaced the failing one-liner data consumption `SqlDataAdapter.Fill(DataSet)` with a loop of `SqlDataReader.GetValue()`.

I tried to make the resulting program as generic/configurable as I could (think of), without wasting too much time on it.

How it works

The application connects to a *(configurable)* MS SQL server. It fetches the schema of a *(configurable)* database and exports **all** data of **all** tables to text files (one file per table) into a *(configurable)* folder. The file names created for each database table follow the format `<schema>.<table>.csv`, e.g. `dbo.CustomerDat.csv`.

Configuration XML

MsSqlDataToText retrieves all necessary configuration data form a single XML file passed as a parameter, e.g. `MsSqlDataToText.exe MyConfiguration.xml`.

The file is divided in four configuration sections, `Database`, `Export`, `SkipColumns` and `TableSelect`, which control the actions taken by MsSqlDataToText.

Section **Database**

- *ConnectionString*
  Defines the connection to the *(source)* MS SQL server. This is literally the ADO.NET connection string.

- *TableQuery*
  This SELECT statement retrieves the schemas/tables of which data should be exported. The query `SELECT schemas.name as schemaName, tables.name as tableName FROM sys.tables INNER JOIN sys.schemas ON tables.schema_id = schemas.schema_id ORDER BY schemas.name, tables.name]` selects all schemas and tables, i.e. all data from the database.

Section **Export**

Controls export behavior:

- *DestinationPath*
  The folder in which the resulting CSV files should be created. ***Please note***: make sure there's enough space for the resulting text files.

- *SkipEmptyTables*
  If enabled (1), for tables with no data (=no rows) in the source database **no** text file will be created.

- *ColumnDelimiter*
  Specify the character that should be used as a column delimiter in the text file. Typically for *(german)* CSV files that's a `;` *(semicolon)*.

- *ColumnNameAsFirstLine*
  Should the first line of each created text file consist of the database column names? Yes (1) or No (0).

Section **SkipColumns**

MsSqlDataToText doesn't handle binary column data types such as `IMAGE`, `BINARY` properly partly due to the fact that I can't know what that binary stream is supposed to be. An image, a document, an email? So here's a way to exclude specific columns or even whole tables from the export.

List any column which shouldn't be exported in the format `<schema>.<tablename>.<column>`, e.g. `dbo.Customerdata.ID`. If a whole table should be skipped altogether, use `*` *(asterisk)* as the column name, e.g. `dbo.Customerdata.*`.

Section **TableSelect**

Instead of exporting all data from each table, this section let's you define specific columns to export, omitting unnecessary/unwanted data that way and potentially speed up the export.

List any individual SELECT statement for a certain table in the format `<schema>.<tablename>|<SELECT columns part>`, e.g. `dbo.Customerdata|ID, LastName`.

To define the default SELECT statement, use `*` *(asterisk)* as the table name, e.g. `*|*`. This is also the default, if not set otherwise here, resulting in a query like `"SELECT * FROM <table>"`.

Here's a sample XML:

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<Configurations>
   <Configuration Default="true" Key="Default" LastRead="1899-12-30T00:00:00.0000"
LastWrite="2019-02-14T10:18:38.0000">
      <CfgSections>
         <CfgSection Key="Database">
         <!--
         Database connection, define the ConnectionString property for the .NET
SqlClient.
         For connection string examples, see
https://www.connectionstrings.com/sql-server/
         -->
            <CfgEntrys>
               <CfgEntry Key="ConnectionString" IsBinary="false" VarType="8">
                  <![CDATA[Server=sqlserver.domain.tld;Database=MyDatabase;User
Id=sa;Password=password;]]>
               </CfgEntry>
               <CfgEntry Key="TableQuery" IsBinary="false" VarType="8">
               <!-- Define which tables to export, this query (the default)
selects all tables = exports all data of those tables -->
                  <![CDATA[SELECT schemas.name as schemaName, tables.name as
tableName FROM sys.tables INNER JOIN sys.schemas ON tables.schema_id =
schemas.schema_id ORDER BY schemas.name, tables.name]]>
               </CfgEntry>
            </CfgEntrys>
         </CfgSection>
         <CfgSection Key="Export">
         <!-- Configure the data export. -->
            <CfgEntrys>
               <CfgEntry Key="SkipEmptyTables" IsBinary="false" VarType="3">
               <!--  Skip the creation of export files for empty (=no data) tables
altogether?, 1 = True (Skip), 0 = False (Don't skip) -->
                  <![CDATA[0]]>
               </CfgEntry>
               <CfgEntry Key="DestinationPath" IsBinary="false" VarType="8">
               <!--  Set a folder as a destination -->
                  <![CDATA[C:\DATA\Exports\]]>
               </CfgEntry>
               <CfgEntry Key="ColumnDelimiter" IsBinary="false" VarType="8">
               <!-- Character used to delimit columns in result text file  -->
                  <![CDATA[;]]>
               </CfgEntry>
               <CfgEntry Key="ColumnNameAsFirstLine" IsBinary="false" VarType="3">
               <!--
               Output the column names as the first line?, 1 = Yes, 0 = No
               Defaults to 1 = Yes, if missing
               -->
                  <![CDATA[1]]>
               </CfgEntry>
            </CfgEntrys>
         </CfgSection>
         <CfgSection Key="SkipColumns">
         <!--
```

```
            Do not export the data of the following columns.
            -->
               <CfgEntrys>
               <!--
               List any column which shouldn't be export in the format <schema>.
<tablename>.<column>, e.g. dbo.Customerdata.ID
               If a table should be skipped altogether, use '*' as the column name,
e.g. dbo.Customerdata.*
               -->
                  <CfgEntry Key=">ColumnName" IsBinary="false" VarType="8">
                     <![CDATA[dbo.Rn_Interaction_Attachment.*]]>
                  </CfgEntry>
                  <CfgEntry Key="ColumnName" IsBinary="false" VarType="8">
                     <![CDATA[dbo.Rn_Interaction_Email.*]]>
                  </CfgEntry>
                  <CfgEntry Key="ColumnName" IsBinary="false" VarType="8">
                     <![CDATA[dbo.Mail_Merges.*]]>
                  </CfgEntry>
               </CfgEntrys>
            </CfgSection>
            <CfgSection Key="TableSelect">
            <!--
            Define individual SQL SELECT statements for tables.
            Only list the column part after SELECT and up until FROM, e.g. for
"SELECT ID, LastName FROM dbo.CustomerData"
            use "ID, LastName"
            -->
               <CfgEntrys>
               <!--
               List any individual SELECT statement for a certain table in the format
<schema>.<tablename>|<SELECT columns part>, e.g. dbo.Customerdata|ID, LastName
               To define the default SELECT statement, use '*' as the table name,
e.g. *|*
               The default column select is "*", i.e. "*|*" will result in "SELECT *
FROM <table>"
               -->
                  <CfgEntry Key="SQLSelect" IsBinary="false" VarType="8">
                  <!-- The default -->
                     <![CDATA[*|*]]>
                  </CfgEntry>
               </CfgEntrys>
            </CfgSection>
         </CfgSections>
      </Configuration>
   </Configurations>
```

Installation

None - simply extract the contents of file `MsSqlDataToText.rar` into a directory, edit the included sample configuration accordingly and start the application.

Known limitations

... too many probably. Most notable though the inability to handle binary column data types such as `IMAGE`. Hence the possibility to exclude those.