Appendix: Generic Database Datastore

2012-10-23

Overview

The generic database datastore can be used to provide general access to database tables and views, for example with the `ReadTableFromDataStore()` command. The trade-off is that although tables and views can be accessed, there is no application programming interface (API) to deal with the intricacies of the database and converting tables to more complex data objects like time series. Consequently, subsequent processing must operate on the returned tables.

The datastore internally corresponds to an Open Database Connectivity (ODBC) connection. The connection can be defined one of two ways:

- Define an ODBC connection using Windows tools. The advantage of this approach is that database authentication occurs through the ODBC connection. The disadvantage is that the connection may use a generic database driver that does not perform as well as vendor drivers. This approach is used when the `DatabaseEngine` and `OdbcName` configuration properties are defined for the datastore.
- Provide connection information via `DatabaseEngine`, `DatabaseServer`, `DatabaseName` (and potentially login configuration properties) and allow the software to use a vendor-specific JDBC (Java Database Connectivity) driver, which is generally optimized for the database software. The disadvantage of this approach is that advanced authentication interfaces have not been implemented (this may or not be an issue depending on the security enabled for the database).

Limitations

The following limitations apply to the generic database datastore:

- Database permissions control which tables and views are accessible and consequently protected tables may not be visible in software or may generate errors if attempts are made to manipulate outside of permissions.
- An attempt is made in the `ReadTableFromDataStore()` command to list tables and views for selection. However, the ability to filter out system tables is limited because some database drivers do not implement required functionality. For example, the SQL Server JDBC driver does not allow generic filtering of system tables and a work-around has been implemented to remove known system table and view names from lists displayed to users.
- Table column properties in TSTool are determined from database column metadata. Although support for common data types has been implemented, some data types may not be fully supported. If a database column type is not supported, the default is to translate the column data to strings in the output table.
- Although database column properties can specify the width and precision for floating point data, some database metadata is inaccessible, causing data-handling or visualization issues. For example, the SQL Server metadata defaults result in the precision of floating point numbers (called “precision” in TSTool and “scale” in SQL Server column properties) to be set to zero.

The work-around is that any floating point data column that has a precision of zero is treated as having a precision of 6 digits after the decimal point.

Datastore Configuration Files

A datastore is configured by enabling the datastore in the main *TSTool.cfg* configuration file, and creating a datastore configuration file for each connection. Configurations are processed at software startup to enable datastores. An example of the TSTool configuration file is shown below. Multiple datastores can be defined using the `[DataStore:DataStoreName]` syntax. Properties for each datastore are specified in an accompanying configuration file described after the following example.

```
# Configuration file for TSTool

...properties omitted…

# Startup datastores (note that datastore name in config file takes precedence)

[DataStore:SomeDatabaseDataStore]
ConfigFile = "SomeDatabaseDataStore.cfg"
```

TSTool Configuration File with Generic Database Datastore Properties

The following illustrates the generic database datastore configuration file format, which in this example is located in the same folder as the TSTool configuration file.

```
# Configuration information for "SomeDatabaseDataStore" datastore (connection).
#
# The user will see the following when interacting with the datastore:
#
# Type – GenericDatabaseDataStore (required as indicated)
# Name - database identifier for use in applications, for example as the
#     input type/name information for time series identifiers (usually a short string)
# Description - database description for reports and user interfaces (a sentence)
# Enabled – whether the datastore is enabled (default=True)
#
# The following are needed to make the low-level data connection:
#
# DatabaseEngine - the database software (SqlServer)
# OdbcName – ODBC name (specify this OR the following properties)
# DatabaseServer - IP or string address for database server
# DatabaseName - database name used by the server
# SystemLogin – the login to be used for the database connection
# SystemPassword – the password to be used for the database connection
#
# Property values can use the notation "Env:xxxx" to use an environment variable,
# "SysProp:xxxx" to use a JRE system property, or "Prompt" to prompt the user for
# the property value (system console is used – not suitable for TSTool startup from
# the Start menu)

Type = "GenericDatabaseDataStore"
Name = "SomeDatabaseDataStore"
Description = "Database on some server"
Enabled = True
DatabaseEngine = "SqlServer"
# Specify OdbcName…
OdbcName = "OdbcName"
# Or, specify the following…
DatabaseServer = "ServerName"
DatabaseName = "DatabaseName"
SystemLogin = "LoginForConnection"
SystemPassword = "PasswordForConnection"
```

Generic Database Datastore Configuration File

The `DatabaseEngine` can be one of the following values, and is used to control internal database interactions:

- `Access` – Microsoft Access database
- `H2` – H2 database
- `Informix` – INFORMIX database
- `MySQL` – MySQL database
- `Oracle` – Oracle database
- `PostgreSQL` – PostgreSQL database
- `SQLServer` – Microsoft SQL Server database

TSTool, which is written in Java, is distributed with the software drivers for the above databases only if datastores have been implemented that use a database product. For example, the State of Colorado's HydroBase database is implemented in SQL Server and consequently the SQL Server driver is distributed with TSTool. Other drivers (e.g., Access via ODBC) depend on installation of the database software, which typically includes the ODBC drivers. Some of the databases listed above have only been used in development and software support may be out of date. If in doubt, contact the software developers and the issue will be evaluated. More databases can be supported if the number of users increases.

The following example illustrates how to configure a datastore for an ODBC DSN connection to an Access database:

```
# Configuration information for Nebraska DNR development database.
# Properties are:
#
# The user will see the following when interacting with the datastore:
#
# Type - required to be GenericDatabaseDataStore
# Name - datastore identifier used in applications, for example as the
#     input type information for time series identifiers (usually a short string)
# Description - datastore description for reports and user interfaces (short phrase)
# DatabaseEngine - the database software
# OdbcName - the Open Database Connectivity Data Source Name (ODBC DSN), configured
#     in Windows Control Panel
#

Type = "GenericDatabaseDataStore"
Name = "ElSalvador"
Description = "El Salvador HydroBase"
DatabaseEngine = "Access"
OdbcName = "ElSalvador"
```

Generic Database Datastore Configuration File Using ODBC DSN

The following example illustrates how to configure a generic datastore for a SQL Server database, using separate database connection properties (NOT using an ODBC DSN). Such configurations may not be suitable because it may be desirable to configure login information in an ODBC DSN. The following is appropriate if a generic read-only service account is configured.

```
# Configuration information for Nebraska DNR development database.
# Properties are:
#
# The user will see the following when interacting with the datastore:
#
# Name - datastore identifier used in applications, for example as the
#     input type information for time series identifiers (usually a short string)
# Description - datastore description for reports and user interfaces (short phrase)
#

Type = "GenericDatabaseDataStore"
Name = "NDNR-Cascade-WaterRights"
Description = "INSIGHT Development Database"
DatabaseEngine = "SqlServer"
DatabaseServer = "xxxxx"
DatabaseName = "WaterRights"
SystemLogin = "guest"
SystemPassword = "guest"
```

Generic Database Datastore Configuration File Using Database Connection Properties