DataService production prototype. (Grillon)

Sean Munson - April 27 2010

The Grillon ("Cricket") code generation application is designed to produce a software package containing the majority of input/output tasks that go on between applications and databases. It is not designed to be a comprehensive ro specific solution; it is designed to create a solution to a general class of tasks that regularly need to be performed. Its aims are to provide :

fast initial deployment,

consistent modeling

strong property typing

flexibility in datastorage selection

reproducible & auditable production processes

Additional future benefits can include;

bounds-checking

relationship constrint checks

Input sanitization & checking

The program is designed to be modular. Each component is high specialized, with the Windows application orchestrating the process and providing the user interface. The prototype is build with several consistent themes;

1. The Commonality of SQL - SQL is variable across multiple DMBS packages, but sticks to a core structure; Schemas are made of tables, tables contain fields, fields are charactarized by types of various charactarizations. A set of charactarization types and sizes should be enough to permit a package to sufficiently describe any database in terms that are therefore interpretable to any other database or - more importantly - most languages.

2. Commonality of lanuage - many modern programming languages have similar structures and abstractions;libraries are composed of sets of classes, which are composed of methods and properties. If a component could interpolate the data types in the SQL.

So using these assumptions, Grillon divides the task of DataLibrary construction into several tasks :

1. Database discovery - Information about the tables, views, and schema should be gatherable

2. Composition of abstract model - an in-application set of objects is constructed

3. Configuration of production process - the settings of what objects and what methods should be produced

4. Database Code generation - The Appropriate programmatics for the target DBMS should be produced

5. Production of code - The Actual Generation of output code.

Each of these is has specific components :

Model Composition: & Database Code generation - IQueryBuilderClass

Configuration of the abstract modeling: Persistence Classes - Table, Database, IxmlSerialize

Production of code : CodebuilderClass

Database Discovery : These components provide connectivity to the target database. They work together to gather information about the database including the names of tables and views, their keys, and the types and sizes of their fields.

Datasource Class - a Class used by the UI to communicate with the Database for interrogation

IDataSourcePlugIn - and interface for components that provide data discovery services. Presently has three implementors;

AccessDataSourcePlugin - Providing data discovery and connection services for MS Access

MySqlDataSourcePlugin - Providing data discovery and connection services for MySql Databases

SqlDataSourcePlugin - Providing data discovery and connection services for Sql Databases

Note: Each of these applications are quite distinct, and it can be expected that components for Oracle and DBIV would be equally complex. Each also contains references to the component libraries required. For example; MySql uses the .net components developed by the authors of MySql.

DataSourceType Enumeration - an enumeration used by the DataSourceclass listing the available data sources.

Modeling and Persistence: The following classes are used to maintain the model, persist it, and are used by the other in lieu of direct access to the database. This gives the application the advantage of being able to work on a theoretical or disconnected database once the data scheme has been acquired or created.

Table: Name of a table, its database connection, primary key data, field list and type, and build configuration.

DataBaseServer: Connection string and name of the Database(s) configured

GenerationSettings: Reserved for Future use

IXmlSerialize : Interface involved in the serialization of the above classes.

Database Code Generation : These components are expected to be able to generate the SQL statements for performing specific functions in the application. They also are responsible for charactarizing the server-specific data into more abstract data charactarizations.

QueryBuilderEncodingType - Provides the abstract data charactarizations (date, string, integer, real, etc)

IQueryBuilder - the interface to which each of the implementations must comply

AccessQueryBuilder - Provides syntactic expertise for data i/o processes using MS Access

MySqlQueryBuilder - Provides syntactic expertise for data i/o processes using MS Access (incomplete)

SQLQuerybuilder - Provides syntactic expertise for data i/o processes using MS Access (incomplete)

Note : In the event that an implemented needed to construct Stored Procedures, these would be the components in which such construction would happen. The IQueryBuilder interface is expected to be extended soon to indicate wether the component supports Stored procedure calls, or Simple SQL calls, or Both.

Code Generation : These components participate in the generation on the output code :

CodeBuilder - an abstract class which provides the base functionality for the code generation components.

CSharpBuilder - generates the codebase in C#

JavaBuilder - Generates the codebase in Java (incomplete)

Presently, the C# code generation class is the only one that is operational. Its output is worth discussing;

The code that is generated follows the factory design pattern. The user is responsible for providing a new (or the same) connection string and creating a DSFactory object ( this class is provided in all output implementations). This class is then used to produce the other classes - either empty, using the Create\_\_\_\_() group of classes , or with Database Data, using the Load\_\_\_() group of classes.