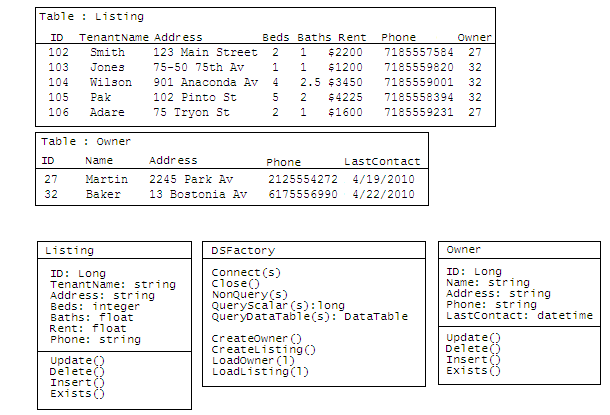
Using documentation for *Grillon*

Overview :

Grillon is an application which is used to generate a starting codebase for applications using a relational database management system. It can be configured to connect to Microsoft SQL servers, MySql Database servers, or even use databases built using Microsoft’s Access product. The Output codebase is presently supports C#, and in the future we expect to provide support for PHP, Java and VB.net.

Grillion is a tool for developing an API of data access classes that interface with a particular database. It provides a connection management class, and a basic method to get record data, alter it, and post it back to the database with a minimum of code. These classes are designed to be strongly typed and present a consistent relationship between the library’s object set and the database structure;

For example : in the below database, The Tables List and Owner are filled with existing complex, strongly typed data. In the development of a web page or web service, the programmer would have to create a suite of custom variables to store retrieved data, and then compose updating SQL commands and make the call manually.

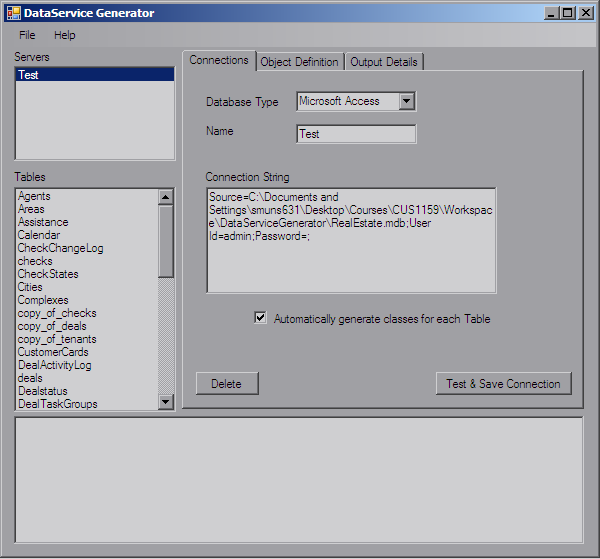
will be onAny change to the database structure, or types or the requirements of the page would require extensive code changes. Moving the same table structure from one type of database to another would require recoding. Addition or removal of fields, or renaming of the fields would require extensive recoding. Names could not be regularly and reliably predicted within the codebase.

The creation of the codebase comparable to that produced by Grillon had taken (not counting changes) upwards of one day per table. In the event of a large relational database containing 100+ tables, this can be several man-months of development time. By automating the production of much of the Database Access code, Grilion can save development time as well as accommodate inevitable changes during the development phase

Requirements:

Grillon runs on a personal computer running Microsoft Windows XP Professional, Vista, or Windows 7. Users should also have network or local access to the database which .

Example Use :

Step 1: Select Database / build Connection :

The first step is to connect to the database that one wishes to use/model. Grillon presently supports Connections to the Microsoft Access Database format through the OLEDB support in the .net Framework. Also supported is the Microsoft SQL server through the SQLClient library in the .net framework. Finally, MYSQL’s .net libraries will permit access to a local or remote MYSQL Server installation. See Appendix 1 for a list of URLs and Connection string Details.

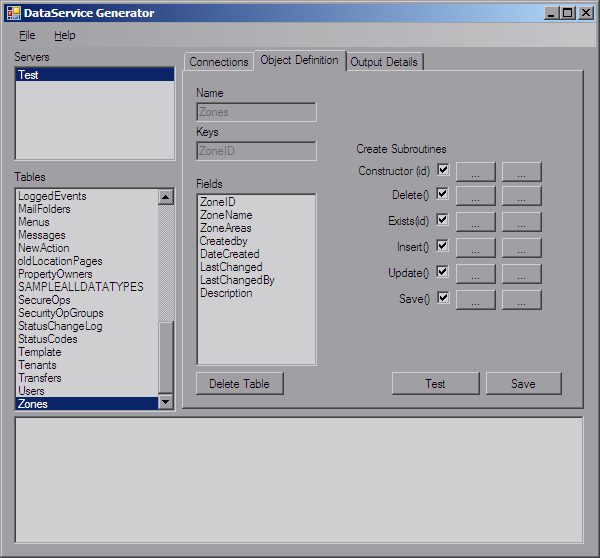
Once the Database Type has been selected , enter the name of the catalog or schema into the Name field. Enter the Connection string value into the Connection string text box. This may provide information such as login or user information, database location or schema selection data.

If you intend to model all available tables, you can select the “Automatically generate classes for each Table” option.

When you click the “Test & Save Connection” button, the application will load the appropriate database plugin, and use the name and connectionstring values to attempt to connect to the database you have indicated. In the event of success, the Name from the Name textbox will appear in the “Servers” listbox. Failure will produce and error message in the output Text box at the bottom of the dialog.

Once the name of your server is in the listbox, you will be set to select the tables and features of your model. Click on the name of your server in the listbox, and the “tables” listbox should fill with all the available Tables that that database contains.

The delete button in the bottom left corner of the tab panel is used to remove a data server from your list of configured servers.

2: Details of your Model ;

Next, define your Dataservice objects. Remember that you will have access to the DSFactory class which will grant you limited ‘raw’ access to the database using SQL strings, but in addition, for each table, Grillon will generate a class for you to instantiate, allowing you much easier, strongly-typed data objects.

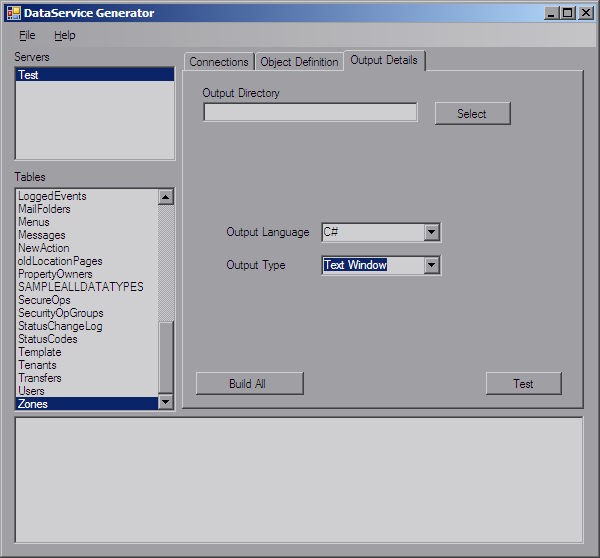
When you clikc on any table in the tables listbox, the “object definition” tab will be brought forward and the details of that object within your build configuration will be displayed

If you had selected the “Automatically generate classes for each Table” option on the previous tab, Every table you select will bring up the name, primary key and files of the table to which the object will be associated. If the table is not configured to have its own class, the fields and keys outputs will be blank. If will notice also that the button labeled “Delete Table” in the diagram above will be labeled “Add table”. Clicking on this will fill the fields list and the Keys textbox.

It is worth nothing that not every deployment will want to have a fully-serviced class for every table. For instance. In many cases, a developer may feel it best to only include the classes he will be using. In addition, he may not wish to implement functions that are not going to be used. Grillon allows you to configure whether each of the subroutines is created.

Directly to the right of these checkboxes are a set of buttons. When clicked these will open a text editor in which the developer may put his own comments or code to run before and/or after the implemented code runs. For example. If the developer wants a custom logging system to record a message before every delete operation, he or she has only to add the code for this call in on the right-hand button following the Delete() checkbox.

Once your production model is fully developed, you should click the File > Save option to store your settings into an XML file. This will allow you to store your work to be loaded into Grillon Later if you wish to alter your codebase or if the data structures have been altered.

Output :

Once you have set up the build instructions of your dataservice, select the “output details” tab and select the output language of your choice (presently, only C# is working) and the output type (presently, all output comes in a text window). Click Build All.

Grillon Will then walk the list of configured objects, and construct a class for each. It will also build an enumeration with the name of each class as an element. It will then create the load() and create() methods for the objects in the DSFactory class to allow developers access to data-enabled objects

When all this is done, the data will be written as a single large text file to the location of the output directory, or to the application’s current working directory if no output directory is configured.

This code can then be integrated into your website or web service.

Integration ;

Integration of the produced code should be simple. Assume that we’re going to make an ASP.net website using c# as our server-side scripting language. We would probably;

1. Create the basic shell Web Application
2. Establish the connection string that we wish to use and put in the web.config <ApplicationSettings > section. This may be different from the connection string used in Grillon to create the Dataservice classes, due to the need to be sensitive to security needs.
3. Instantiate DSFactory when data access begins, and cash it in the Application.Values Collection.
4. Create and use your class objects as needed.

It should be noted that many of the operations of interaction with the database, such as performing searches and listing possible matches will probably still need to be performed with Microsoft’s Data Object libraries. For Example, You may still need to construct a Search SQL command and bind the resulting DataTable to a gridview control in the web page. This level of functionality is supported with the use of the NonQuery(), QueryScalar(), and QueryDataTable() functions of the DSFactory class, which allow access in line with the connectionstring’s user settings.