Annales UMCS Informatica AI XII, 1 (2012) 79–87
DOI: 10.2478/v10065-012-0008-0

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI
http://www.annales.umcs.lublin.pl/

Generating of Business Database Application Elements

Artur Kornatka[1,2*]

[1] *Institute of Computer Science, Maria Curie-Sklodowska University, pl. M. Curie-Sklodowskiej 5, 20-031 Lublin, Poland.*
[2] *Department of Computer Science, Nowy Sącz School of Business - National-Louis University, Zielona 27, 33-300 Nowy Sącz, Poland*

Abstract – The paper presents the outline of an innovative conception of the functioning of generator for business database applications elements and shows also the working principles of the author's prototype system named BACG (Business Application Code Generator) which implements the aforementioned conception.

1 Introduction

The main factor determining the cost of software creation is the value of programmers labour expenses [**1**]. High competition on the software market for business application compels producers to reduce these costs. Substantial reduction of expenses pertaining to implementation of information systems can be achieved through using the specialized generators which enable replacement of programmers and speed up the process of software production. Usage of optimally working generators allows for automation of selected stages of business application developments while at the same time keeping high quality standards of the final product. In this case it is very important that all generated elements differ as little as possible from those created by a programmer. Skilful application of such specialized systems by companies producing business applications, may be a key factor determining their competitive advantage on the market.

The main aim of the paper is to present an innovative concept of functioning of generator for selected code elements of database business application and showing

*akornatka@gmail.com

the principles of working of the author's prototype system called BACG (Business Application Code Generator), which is meant to accomplish this concept.

The BACG system is an innovative generator of business application elements. The main feature which makes it different form the other similar tools ([**2**], [**3**]) is its focus on the generating of optimal and professional code which is coherent with the standing design patterns. The code is created automatically according to the rules and principles obligatory in advanced business applications development, and is almost indiscernible from the one that would be created by a professional programmer. Thus, the BACG system replaces a programmer during development of advanced code in contrast to the existing systems which concentrate on the generating of suitable forms only without paying attention to the quality of the built code.

The BACG system was created with the help of the Microsoft Visual Studio 2010. The application code was written in the C# language. The generator is able to work with the databases managed by the Microsoft SQL Server 2008.

All code elements which have been generated by the BACG system are compatible with the MVVM design pattern. The second section of the present work contains description of this template and technology used in the generated parts of application.

The third section describes the process of generating of selected business application elements.

Theresearch results are presented in section 4 where the innovative features of the BACG system have been gathered and further development of the system is presented.

2 Theory – short description of MVVM, WPF, and XAML

WPF (Windows Presentation Foundation) is a presentation system for building Windows client applications with visually stunning user experiences. It defines the latest programming standard for the expanded user interfaces. Due to this technology the programmers can use up-to-date controls which allow to employ the full possibilities that are offered by the Microsoft operating system.

XAML (eXtensible Application Markup Language) is a declarative markup language created by Microsoft for programming user interfaces created in the WPF technology. The syntax of XML is based on the classical XML language. Subsequent tags of the XAML language describe all elements of the WPF user interface.

More details on WPF and XAML can be found in [**4**].

MVVM (Model-View-ViewModel) is a modern design pattern used for creating applications with the extended presentation part. The main purpose of this pattern is the separation of three basic layers of the application. According to [**5**] these are the Model, View, and ViewModel layers.

The Model layer of MVVM describes data logic or business logic of the application. This layer is completely independent of the user interface. It consists of many business objects which implement specific goals of the application.

The View layer of MVVM consists of visual elements of the application. A view may be an application window or the user control (UserControl) which can be placed on any application window.

The ViewModel layer of MVVM makes up a connection between a model and a view. The main task of objects in this layer is retrieving the selected data from a source, next transforming them into the form characteristic of the given view, or passing the properly modified information from the view to the data source.

The MVVM design pattern is ideally suited for creating user interfaces with the help of the WPF technology.
More information on MVVM can be found in [**6**], [**7**], [**8**].

3 The process of generating of business database applications

Let us assume that using the Microsoft Visual Studio 2010 development environment we want to create a professional business application. The programming language we select is C#. We also assume that the database of this application has been created in Microsoft SQL Server 2008. The database should contain all necessary tables and relationships between them. After all necessary parameters for setting up a connection with the database have been introduced, we can set about to apply the BACG system in order to generate the elements of business application which is being created. After launching, the BACG system asks for providing the name of a directory where all generated elements will be stored, next it retrieves information about the structure of selected tables from the database server and displays it on the screen.

The process of generating of database business application elements by the BACG system can be divided into five main stages. It starts with the generating of the elementary classes and fundamental stored procedures. Next, the system creates subsequent elements compatible with the above described design pattern MVVM. Thus BACG generates the Model layer, that is the classes which are responsible for contact with the database. The next stage is the creation of the View layer, that is advanced views realizing specific scenarios. In this stage the components of the ViewModel layers are also generated which are meant for intermediation between the Model and View layers.

After this cursory description of operating of the generator, we can move on to the detailed description of the aforementioned parts.

3.1 Elementary classes generation

In the first stage of generating of database business application elements the BACG system creates a collection of elementary classes. During this stage a collection of classes which facilitates the basic operations comes into being.

The first element to be generated is the AccessToDataBase class. It contains a private field named connectionString which keeps all parameters necessary for setting up the connection with SQL Server and the database created on this server. A constructor

of this class initializes this field. The key element here is the method named CreateConnection() which creates an open connection with the database (on the server) and returns an object of the SQLConnection type.

In this stage a class named ObjectQuery is also created. It contains a set of static methods responsible for calling SQL queries and various stored procedures which are located on the database server. An example of such method can be a function named RunSingleValueProcedure(String procedureName) which runs the procedure (passed to it by its name) returning single value.

The next class which is generated in this stage is named DelegateCommand. It enables calling the indicated function specified in the class layer ViewModel by the element defined in the layer View. All of this is accomplished through the Command mechanism, which is supported by the WPF technology.

This stage is finalized by the creating of the ComboBoxKeyAndValue class which is necessary for the correct functioning of the controls ComboBox on the View layer views.

3.2 Stored procedures generating

In the subsequent stage of the generator run we can choose the tables which – in the future application – will be subjected to such operations as: adding, deleting, retrieving all or selected records. For each of these tables BACG generates a suitable SQL query code which performs the mentioned operations. Then on the basis of each such SQL query, the SQL code creating the stored procedure on the server is formed. BACG sends and executes this code on the SQL Server. As a consequence of this process, a set of stored procedures on the database server comes into being which enables addition, deletion, retrieval of all or selected records. Of course, these procedures are generated for all tables indicated by the user of the system.

3.3 Generating of the classes responsible for the database operations

After the elementary classes and stored procedures have been created, the generator proceeds to the creation of layer Model elements according to the MVVM design pattern.

3.3.1 "Type R" classes

The entity class – which will be called "type R" – is created for each table from the collection selected in section 3.2. During the construction of these classes a popular C# mechanism of properties is used. Hence, for each table field a property is created in the corresponding "type R" class, which has the same name and associate type.

The creation of every property is accompanied by generating two methods: get and set which allow reading and setting values to specified private fields of the class. The set method contains a mechanism for checking whether a new value is the same as the existing one. In this case a new assignment is not carried out. During the "type R"

class generation, a special attention should be paid to the table fields which are foreign keys (connected with some record from the related table). In this case, apart from the standard field and property, an additional field is created (and the corresponding property) in the "type R" class and its type is determined by the "type R" class of the related table. In this case, apart from the access to the foreign key values, we can obtain a direct access to the related object. Calling the get method for such a field (of related object) is bound up with running on the database server a suitable stored procedure which takes data from the related record. Data are assigned to the object and the field becomes a reference to this object.

3.3.2 "Type C" classes

When the "type C" class generation is completed, the BACG system starts generating classes which are responsible for several database operations. Thus for each table selected in point 3.2 we can create a class which will enable us to add, delete, update, or read records from this table. We will call such classes "type C".

Here are some exemplary functions contained in the "type C" class:

- the public method Add(...) to which an object of the "type R" class is passed, and then by calling a proper system procedure of the database server it adds a new related to this object record in the table
- the public method Delete(...) which calls a proper stored procedure to delete a specified database record
- the public method GetAll(...) which returns a collection of "type R" classes; this method calls a stored procedure retrieving all records form the selected table, and next it creates the proper object of the "type R" class from each record and adds it to the objects collection.

3.4 Views generation

The next stage of the database business application elements generation by the BACG system is the creation of views, that is the elements of the View layer according to the MVVM design pattern. Views are created with the help of the WPF technology and XAML language.

A user of the BACG system picks up a table for which he would like to create the views, and next he invokes a special window of the generator. In this window BACG displays all fields of the selected table together with their properties. Apart from that it presents tables related to the one selected. After the selection of the foreign key we see all fields in the related table. Moreover, the window allows us to choose the scenario of the object which is being created.

Currently the BACG system generates views according to two scenarios.

3.4.1 The first scenario – ObjectVew

The BACG generator is capable of creating a view according to the first scenario. This scenario allows to save, edit, and update the selected record in the table stored in the database – which is of the "type R" class object. For example, if we want to create a view allowing to save a new record in a table, first we point to the fields which are to be filled. According to the WPF rules the BACG system, with the help of the XAML language, creates a special UserControl and for each pointed field adds a label describing this field (usually a component of the Label type), and next adds the editable field (e.g. TextBox or DataPicker). UserControl is associated with a suitable object of the ViewModel class and each editable field is assigned to the suitable property defined in the class of this object (see section 3.5). Fields association is performed through the Binding mechanism which is supported by the WPF technology. All labels and fields are placed in a special component Grid which controls a displacement of these elements. However, the fields corresponding to the table foreign keys are treated in a special way. For example, for such fields there can be created a special ComboBox which is able to display the related records from the related table so instead of filling in the value of the foreign key one can choose from the expandable list.

The UserControl element contains also the button which invokes a save function from a suitable ViewModel class. This action is realized by the Command of the Button control property.

Created UserControl can be placed in any window of the business application.
subsubsectionThe second scenario – ShowAllView The BACG generator is able to create a view according to the second scenario. This scenario allows to present all records from the indicated table or all objects which are returned by the function GetAll() called for an object of the chosen "type C" class. Similarly to the previous case each view is a separate UserControl. The DataGrid component (available in WPF) is responsible for displaying objects in a table. The whole UserControl is associated witch a suitable object of a specially created ViewModel class (see subsection 3.5). Further, DataGrid has been associated with a suitable collection of objects defined in this class. The Binding mechanism has also been used in this case. The DataGrid type column is generated depending on the field type displayed in this column. Instead of showing the foreign keys the suitable data from the related record/object are presented. The user decides about which data are displayed.

3.5 Class ViewModel generation

When subsequent views are generated, the BACG system creates the classes of the ViewModel layer. The main task of these classes is providing data in a suitable form to the views or passing from the view properly modified information to a data source. These classes act as intermediary between the View layer elements (created in the WPF technology) and the Model layer classes. In the same way as with the views, which are generated by two scenarios, we can also divide the ViewModel layer classes into two categories. The first category includes the classes created for the views produced by

the first scenario, the second one includes those for the views produced by the second scenario.

3.5.1 The ViewModel class generated for the views according to the first scenario

All classes of the ViewModel, generated for the views according to the first scenario, contain two fields. One field is an object of the suitable "type R" class, and the second field is an object of the related "type C" class. The "type R" and "type C" classes are those that have been created for the tables on which the view is to act upon. Additionally, the ViewMoodel layer classes contain a constructor which initializes their files based on its own parameters. The key element of these classes is the Properties area which contains a collection of properties which correspond to the editable controls of the view. Of course, these controls are bound to these properties through the Binding mechanism. The get method obtains a suitable field of the above described object of the "type R" class while th eset method assigns a value to it. The generator pays attention to the consistence of the types between the "type R" class properties and the class ViewModel properties.

Special properties have been generated for the elements of the ComboBox type views which return a list of ComboBoxKeyAndValue type objects. These properties, using their own get method (), call a specially generated function GetAllOnlySelectedFields_FieldsName() for a "type C" class object, which returns the pointed collection. Thus, the generator has to complete the code of the suitable "type C" classes with new functions. In consequence, it requires to create new stored procedures which are to provide selected data to the returned objects collection of the ComboBoxKeyAndValue type.

The last elements of these classes are special properties (ICommand type) which in the get() method create a command of the DelegateCommand type by calling suitable methods for the "type C" class object. As an example we can mention here the property public ICommand SaveCommand which creates a new object of the DelegateCommand class in the method get() by calling the method Add() for the "type C" class object.

3.5.2 The ViewModel class generated for the views according to the second scenario

Classes of the ViewModel, generated for the views according to the second scenario, contain one field. It is an object of the related "type C" class. Moreover, they contain the initializing constructor of the class field, based on its own parameter. The key part of these classes is the region Properties where the property named Show is located, which in the get() method returns a list of objects of the specially defined ObjectNameForAllView type by calling a new function getForAllView() for an object of the "type C" class. The ObjectNameForAllView type is a separately generated class which is composed only of properties created on the fields that a user would like to see in the view that is being created. In this situation it is necessary to complete a suitable "type

C" class with the getForAllView() method I whose purpose is creating a collection of objects of the ObjectNameForAllView type and filling in the data in these objects. Of course, in this case also the retrieving of suitable data from a database is bound with the calling the proper stored procedures kept in the SQL Server database management system.

The Show property is associated in the view with a DataGrid component and its subsequent columns with subsequent properties of the new ObjectNameForAllView type.

4 Summary

The BACG generator is an innovative system which automatically creates selected code elements of the database business applications. Its innovative character is realized through the following features:

- the system generates the optimal, professional source code using up-to-date advanced programming technologies and binds it with the business presentation layer (in contrast to the other generators which focus mainly on the forms creation without paying attention to the code quality),
- generated elements of the business application are compliant with the current Model View ViewModel (MVVM) design pattern,
- owing to the application of the readable design pattern a programmer retains a full control over the automatically generated code, and when the needs arise it can be easily modified and completed,
- each layer of the generated application is created independently, hence it is easy to modify one layer without interfering with the others (e.g. in the case when the user interface has to be changed),
- all generated views are created with painstaking attention to their business functionality and future application for the more elaborate generator,
- the BACG system uses up-to-date technologies for building the advanced Windows forms,
- the BACG system guarantees an effective code which is responsible for the access to the database (the Model layer) by strict binding with the stored procedures located on the database server.

In the future development of the BACG system the author plans:

- to extend the module for view generating so that it will enable to create more professional forms meeting the sophisticated business demands,
- to add a mechanism for extracting the interfaces and abstract classes in the process of code generation for specialized business functions,
- to create a mechanism enabling a cooperation with the object query language, operating on the objects of the Model layer which will be tightly bound with the generator of the suitable stored procedures kept in the database servers.

References

[1] Pressman R. S., Software engineering: a practitioner's approach, McGraw-Hill, 2005.
[2] Patrick T., Microsoft ADO.NET 4, Microsoft Press, 2010.
[3] Visual Studio LightSwitch, MSDN, http://msdn.microsoft.com/pl-pl/library/ff851953.aspx
[4] Nathan A., Windows Presentation Foundation, Sams Publishing, 2007.
[5] Gossman J., Introduction to Model/View/ViewModel pattern for building WPF apps, MSDN, http://blogs.msdn.com/b/johngossman/archive/2005/10/08/478683.aspx
[6] Garofalo R., Building Enterprise Application with Windows Presentation Foundation and the Model View ViewModel Pattern, O'Reilly, 2011.
[7] Smith J., WPF Apps With The Model-View-ViewModel Design Pattern, MSDN Magazine, http://msdn.microsoft.com/en-us/magazine/dd419663.aspx
[8] Sorensen E., Mikailesc M., Model-View-ViewModel (MVVM) Design Pattern using Windows Presentation Foundation (WPF) Technology,
http://megabyte.utm.ro/articole/2010/info/sem1/InfoStraini_Pdf/1.pdf