

## АВТОГЕНЕРАТОР КЛАССОВ

Е. С. Малахова

Институт кибернетики Национального исследовательского  
Томского политехнического университета, 634034, Томск, Россия

---

УДК 004.021

Рассматривается один из многочисленных подходов к генерации кода классов на языке C#. Разработано приложение, позволяющее создавать классы, конструкторы, свойства и методы на основе шаблонов.

**Ключевые слова:** шаблоны, класс, таблица, база данных, атрибут, свойство, метод, конструктор.

The article describes one of the many approaches to generate code of classes in C#. It was developed as an application that allows to create classes, constructors, properties, and methods based on templates.

**Key words:** templates, class, table, database, attribute, property, method, constructor.

**Введение.** В настоящее время на любом предприятии имеется информационная система. Основу такой системы составляет база данных (БД), в которой хранится информация за различные периоды и по разным направлениям деятельности. В процессе автоматизации работы зачастую необходимо использовать объектно-ориентированный подход, анализировать структуру и содержание баз данных.

Для обеспечения работоспособности системы прежде всего необходимо создать большое количество классов для работы с данными, источником которых является БД. Написание кода различных классов может требовать больших временных затрат, поэтому возникла необходимость разработки и реализации алгоритма генерации кода классов. Возможность разработки такой программы обусловлена тем, что при создании классов программист выполняет однотипные операции, в результате чего код классов различается только названием полей. Для оптимизации использования рабочего времени, затрачиваемого на повторение одного и того же алгоритма, было создано приложение, позволяющее автоматически сформировать код класса, используя файлы-шаблоны, в которых содержатся определенные правила оформления классов.

Ниже приведены основные этапы реализации приложения:

- 1) создание метода для соединения с БД;
- 2) получение данных о структуре БД;
- 3) создание алгоритма создания класса;
- 4) сопоставление типов данных;
- 5) разработка и реализация алгоритма генерации свойств и конструкторов.

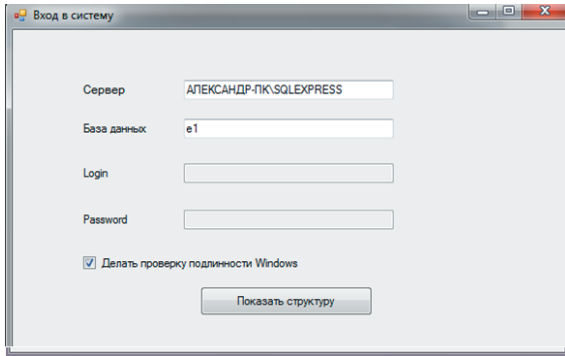


Рис. 1. Форма "Вход в систему"

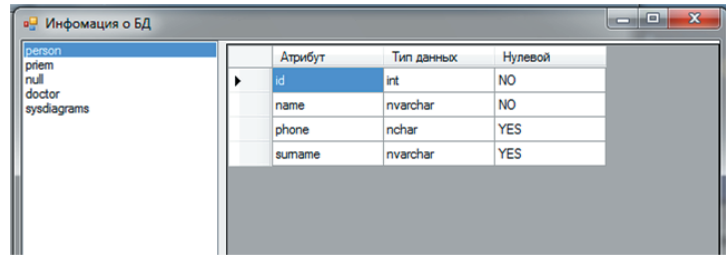


Рис. 2. Форма "Информация о БД"

**1. Метод соединения с БД.** Для удобства пользователей разработан графический интерфейс (рис. 1), позволяющий указать необходимую БД, а также всю используемую для соединения информацию на определенном сервере.

Пользователь вводит информацию о сервере БД, имени БД, а также имеет возможность выбрать способ аутентификации. При проверке подлинности Windows отсутствует необходимость вводить логин и пароль, в остальных случаях их необходимо ввести. Если пользователь ввел некорректные данные, система предложит повторить ввод. Для построения строки соединения с БД используется класс `SqlConnectionStringBuilder()` – построитель строк подключения, позволяющий программным способом создавать синтаксически правильные строки подключения, а также анализировать и перестраивать существующие строки подключения с помощью свойств и методов этого класса [1]. Построитель строк подключения предоставляет строго типизированные свойства, соответствующие известным парам ключ – значение, которые разрешены SQL Server. Разработчики, которым требуется создавать или изменять строки подключения как части приложений, могут использовать класс `SqlConnectionStringBuilder`. Класс также упрощает управление строками подключения, которые хранятся в файле конфигурации приложения [1].

Класс `SqlConnectionStringBuilder` выполняет проверки на наличие допустимых пар ключ – значение. Таким образом, этот класс нельзя использовать для создания недопустимых строк подключения; попытка добавить недопустимые пары приведет к игнорированию исключения.

**2. Данные о структуре БД.** При корректном вводе идентификационных данных пользователю отобразится форма "Информация о БД" (рис. 2). В выбранной из списка таблице отобразится полная информация о ее содержании.

Для получения данных о структуре БД использовалась системная таблица `INFORMATION_SCHEMA`, которая обеспечивает доступ к метаданным БД. Метаданные (каталог системы, словарь данных) представляют собой данные относительно данных, имени БД или таблицы, тип данных столбца или привилегии доступа.

`INFORMATION_SCHEMA` – информационная БД, в которой сохраняется информация обо всех других БД, которые хранятся на сервере SQL. Внутри `INFORMATION_SCHEMA` имеется несколько таблиц только для чтения. Фактически это не обычные таблицы, а их представления, так как отсутствуют связанные с ними файлы.

Для размещения информации о структуре таблицы использован класс `DataSet`. Класс `DataSet` представляет собой содержащийся в памяти кеш данных. Класс `DataSet` состоит из кол-

лекции таблиц DataTable, которые можно связать между собой с помощью объектов класса DataRelation [2].

Чтение и запись данных и схем в экземпляр класса DataSet осуществляются так же, как запись и чтение XML-документов. Данные и схемы можно передавать по протоколу HTTP и использовать в любом приложении на любой платформе, поддерживающей формат XML. С помощью метода WriteXmlSchema схему можно сохранить в качестве XML-схемы. Кроме того, схему и данные можно сохранить, используя метод WriteXml. Для чтения XML-документа, содержащего данные и схемы, используется метод ReadXml [2].

**3. Алгоритм создания класса.** Классы – основа каждого объектно-ориентированного языка [3]. Класс представляет собой инкапсуляцию данных и методов их обработки. Это справедливо для любых объектно-ориентированных языков, различающихся лишь типами данных, хранящихся в виде членов, а также возможностями классов. В том, что касается классов и многих функций языка, язык C# отчасти аналогичен языкам C++ и Java, в то же время сделанные в нем усовершенствования позволяют находить более эффективные решения старых проблем.

Класс представляет собой шаблон, который определяет форму объекта [3] и задает как данные, так и код, который оперирует этими данными. Язык C# использует спецификацию класса для создания объектов – экземпляров класса. Таким образом, класс – множество намерений (планов), определяющих, каким образом должен быть построен объект. Следует отметить, что класс – это логическая абстракция, которую не имеет смысла реализовывать до тех пор, пока не будет создан объект класса и в памяти не появится его физическое представление. Методы и переменные, составляющие класс, называются членами класса.

Для создания классов используются шаблоны. В файлах содержатся определенные наборы строк, которые затем формируются в класс. В качестве шаблона для классов использовался файл class.txt:

```
$using$
namespace $namespace$
{
    public class $tablename$ : BasicEntity
    {
        $constructor$

        $property$

        $methods$
    }
}
```

В процессе формирования класса регулярные выражения заменяются на реальные данные:

\$using\$ – на список подключаемых библиотек;

\$namespace\$ – на введенный пользователем namespace;

\$tablename\$ – на название таблицы, которую пользователь выбрал из списка;

\$constructor\$ – на конструкторы класса;

\$methods\$ – на методы класса;

\$property\$ – на шаблон для свойств класса.

**4. Сопоставление типов данных.** Проведем сопоставление типов данных SQL и C# (табл. 1, 2).

Таблица 1

Типы данных в C# [4]

Тип	Область значений	Размер слова
sbyte	От -128 до 127	Знаковое, 8-бит, целое
Byte	От 0 до 255	Беззнаковое, 8-бит, целое
Char	От U+0000 до U+ffff	16-битовый символ Unicode
bool	True или false	1 байт
short	От -32 768 до 32 767	Знаковое 16-бит целое
ushort	От 0 до 65 535	Беззнаковое 16-бит целое
int	От -2 147 483 648 до 2 147 483 647	Знаковое 32-бит целое
uint	От 0 до 4 294 967 295	Беззнаковое 32-бит целое
long	От -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807	Знаковое 32-бит целое
ulong	От 0 до 18 446 744 073 709 551 615	Беззнаковое 32-бит целое
float	От $\pm 1,5 \cdot 10^{-45}$ до $\pm 3,4 \cdot 10^{33}$	4 байта, точность – 7 разрядов
double	От $\pm 5 \cdot 10^{-324}$ до $\pm 1,7 \cdot 10^{306}$	8 байт, точность – 16 разрядов
decimal	–	12 байт, точность – 28 разрядов
datetime	1.01.1753 г. – 31.12.9999 г.	Дата и время, точность – 3/100 с (3,33 мс)
Guid	–	Глобально-уникальный идентификатор

Таблица 2

Типы данных в SQL [5]

Тип данных	Определение MSSQL и пояснения
bigint	8-байтовые целочисленные данные
Bit	Целочисленные данные только со значениями 1 или 0, обычно заменяются константами 'T' и 'F'
Char	Текстовые данные фиксированной длины (не Unicode). Максимальная длина 8000 символов
Datetime	Дата и время с 1 января 1753 г. до 31 декабря 9999 г., точность 3/100 с (3,33 мс)
Decimal	Числовые данные с фиксированной точностью. Диапазон от $-10^{38} - 1$ до $10^{38} - 1$
Float	Вещественные числовые данные. Диапазон от $-1,79 \cdot 10^{38}$ до $1,79 \cdot 10^{38}$
Int	4-байтовые целочисленные данные. Диапазон от $-2^{31}$ ( $-2\ 147\ 483\ 648$ ) до $2^{31}-1$ ( $2\ 147\ 483\ 647$ )
Money	Денежные данные. Диапазон от $-2^{63}$ ( $-922\ 337\ 203\ 685\ 477,5808$ ) до $2^{63}-1$ ( $+922\ 337\ 203\ 685\ 477,5807$ ), точность до 1/10 000 денежной единицы
Nchar	Символьные данные фиксированной длины (Unicode). Максимальная длина 4000 символов
Ntext	Символьные данные переменной длины (Unicode). Максимальная длина $2^{30}-1$ ( $1\ 073\ 741\ 823$ ) символов
Numeric	В СУБД MS SQL decimal и numeric эквивалентны
Nvarchar	Символьные данные переменной длины. Максимальная длина 4000 символов
Real	Вещественные данные. Диапазон от $-3,40 \cdot 10^{308}$ до $3,40 \cdot 10^{308}$
Smalldatetime	Дата и время с 1 января 1900 г. до 6 июня 2079 г., точность 1 мин
Smallint	2-байтовые целочисленные данные. Диапазон от $-2^{15}$ ( $-32\ 768$ ) до $2^{15}-1$ ( $32\ 767$ )
Smallmoney	Денежные данные. Диапазон от $-214\ 748,3648$ до $+214\ 748,3647$ , точность 1/10 000 денежной единицы
Text	Символьные данные переменной длины (не Unicode). Максимальная длина $2^{31}-1$ ( $2\ 147\ 483\ 647$ ) символов
Tinyint	1-байтовое целочисленное значение без знака. Диапазон от 0 до 255
Varchar	Символьные данные переменной длины (не Unicode). Максимальная длина 8000 символов
Uniqueidentifier	Глобально-уникальный идентификатор (GUID)

Таблица 3

## Сопоставленные типы данных

Тип в C#	Тип в SQL
Byte	Tinyint
Char	Char
Bool	Bit
Short	Smallint
Int	Integer (int), smallmoney
Long	Money, bigint
Double	Real, float
Decimal	Numeric, decimal
Datetime	Datetime, smalldatetime
Guid	Uniqueidentifier
String	Nchar, varchar, text, nvarchar, ntext, nchar

На основе табл. 3 созданы шаблоны свойств, учитывающие особенности конвертации типов данных. Например, различаются строки конвертации для данных, которые могут содержать нулевые значения и которые не содержат их.

**5. Метод генерации свойств.** Одно из основных преимуществ языков объектно-ориентированного программирования, таких как C#, состоит в том, что с их помощью можно определять специальные методы, вызываемые при создании каждого экземпляра класса. Эти методы называются конструкторами (constructors). Конструктор инициализирует объект при его создании и имеет такое же имя, что и сам класс, а синтаксически подобен методу. Однако в определении конструкторов не указывается тип возвращаемого значения [3].

Свойство в языке C# включает объявление поля и применяемых для изменения значения поля методов-аксессоров, называемых получателями (getter) и установщиками (setter). Методы-получатели используются для получения значения поля, а установщики – для его изменения. Свойство имеет два стандартных метода-аксессора: get и set [3].

Таким образом, в классе содержатся различные свойства, которые составляют по одинаковой логике. Это позволило создать шаблон конструкторов и свойств. В качестве передаваемого параметра в шаблон конструктора передается название таблицы, которое является также названием класса. В шаблон свойств передаются названия столбцов таблицы и типы данных, в соответствии с типом данных выбирается строка конвертации данных.

Для хранения набора строк конвертации предлагается использовать класс Dictionary. Универсальный класс Dictionary<TKey, TValue> обеспечивает отображение множества ключей в множество значений. Каждый элемент, добавляемый в словарь, состоит из значения и соответствующего ему ключа. Извлечение значения по его ключу происходит очень быстро, со скоростью, близкой к  $O(1)$ , поскольку класс Dictionary<TKey, TValue> реализован в виде хеш-таблицы [6].

До тех пор пока объект используется в качестве ключа в словаре Dictionary<TKey, TValue>, он не должен изменяться таким образом, чтобы это отражалось на его хеш-коде. Каждый ключ в словаре Dictionary<TKey, TValue> должен быть уникальным в соответствии с компаратором словаря, проверяющим на равенство. Ключ не может быть пустым (null), в то время как значение может быть пустым, если его тип TValue является ссылочным типом [6].

Поскольку класс Dictionary<TKey, TValue> лишен возможности сериализации (сохранения в файл и чтения из файла), возникла необходимость создания собственного класса SerializableDictionary, который являлся бы наследником класса Dictionary<TKey, TValue> и содержал интерфейс сериализации. Также был создан класс, содержащий необходимые поля для подстановки в конечный код фактических названий типов данных и строк конвертации.

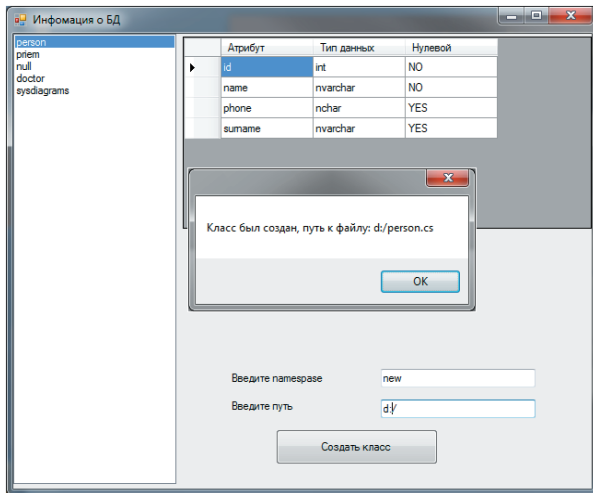


Рис. 3. Форма для создания класса по указанным шаблонам

В качестве визуального результата была разработана форма, позволяющая создать класс по указанным заранее шаблонам (рис. 3). При нажатии на кнопку "Создать класс" срабатывает обработчик события createButton\_Click класса InformationForm. При обработке этого события используется вспомогательный класс ClassBuilder, который и выполняет все основные операции по замене регулярных выражений реальными данными и возвращает готовый код класса, сохраняемый затем в папке, указанной пользователем, или по умолчанию в формате.

**Заключение.** В результате проведенного исследования составлены правила формирования

классов: разработаны файлы шаблонов конструкторов, свойств, методов. Проведен анализ типов данных в языке C# и SQL. На основе проведенного анализа составлена таблица соответствия типов. Разработан алгоритм составления класса из файлов шаблонов для встраивания в соответствующие места шаблонов названий таблиц, атрибутов, типов данных. Составлены строки конвертации для nullable- и notNullable-типов. Создан класс SerializableDictionary – наследник класса Dictionary, который позволяет реализовывать интерфейс сериализации: объекты класса SerializableDictionary можно сохранить в файл, а затем прочесть. Использование внешнего файла с настройками конвертации позволяет, не меняя код программы, изменить правила формирования генерируемого класса. Создан вспомогательный класс ConvertClass, объекты которого хранят информацию из настроечного файла. Разработан графический интерфейс для подключения и работы с необходимой БД, который учитывает специфику соединения с СУБД Microsoft SQL Server 2005.

#### Список литературы

1. MSDN. [Электрон. ресурс]. <http://msdn.microsoft.com/ru-ru/library/xfhwa508.aspx>.
2. MSDN. [Электрон. ресурс]. <http://msdn.microsoft.com/ru-ru/library/system.data.dataset.aspx>.
3. Полный справочник по C#: Пер. с англ. М.: Издат. дом "Вильямс", 2004. 752 с.
4. Типы данных C#. [Электрон. ресурс]. <http://simple-cs.ru/store/csharp/4/>.
5. Типы данных sql. [Электрон. ресурс]. <http://www.firebirdsql.org/manual/ru/migration-mssql-data-types-ru.html>.
6. ЛАБОР В. В. Си Шарп: Создание приложений для Windows. Мн.: Харвест, 2003. 384 с.

Малахова Елена Сергеевна – студентка Института кибернетики  
Томского политехнического университета; e-mail: angel\_of\_dark@tpu.ru

Дата поступления – 02.11.11