第22卷 第8期
2012年8月

计算机技术与发展
COMPUTER TECHNOLOGY AND DEVELOPMENT

Vol. 22 No. 8
Aug. 2012

基于模板的.Net三层架构的代码自动生成

刘于新,姚凯学,许道云
(贵州大学 计算机科学与信息学院,贵州 贵阳 550025)

摘 要:三层架构是数据库开发中最常用的分层架构。三层结构能够提高代码重用率,降低项目开发难度。为了解决手工编写三层架构的代码工作量大且容易出错的问题,使用.Net平台提供的动态编译和反射技术,设计并实现了基于可定制模板的自动代码生成器。该代码生成器利用三层架构的代码依赖数据库的架构信息这一特点,以数据库架构信息、Xml配置文件和模板文件作为输入,输出三层架构的各层代码。用户可以修改Xml配置文件的内容和使用该代码生成器提供的模板语言定制、修改自己的模板文件,方便、灵活地控制输出的目标代码。代码生成器的使用在实际的项目开发中具有重要的意义。

关键词:自动代码生成;三层架构;可定制模板;元数据;内置模板
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2012)08-0013-04

Automatic Code Generation Based on Template on .Net Framework of Three Layers Architecture

LIU Yu-xin, YAO Kai-xue, XU Dao-yun
(College of Computer Science and Information, Guizhou University, Guiyang 550025, China)

Abstract: Three layers architecture is the most commonly used layered structure in database development. Three layers structure can improve code reuse rate and reduce the difficulty of project development. In order to solve the problem that manual programming code of three layers architecture is a big workload and easy to make mistake, using dynamic compile and reflecting technique provided by the .Net platform designs and realizes the automatic code generator that based on the customizable template. The code generator uses this feature that the code of three layers structure relies on the structure of the database information. Taking database schema information, Xml configuration files and template files as input, the code generator produces the code of three layers architecture as output. Users can change the content of the Xml configuration files, and can customize, modify their own template files by using template language provided by the code generator, conveniently and flexibly controlling the output of the object code. In practical project development the use of code generator is of great significance.

Key words: automatic code generation; three layers architecture; customizable template; metadata; built-in template

0 引 言

在MIS系统的开发中,为了提高代码复用率和适应系统界面经常变化的要求,常用到三层架构设计。手工编写三层架构的代码工作乏味而且还存在大量的重复劳动[1,2],这制约着项目的开发进度,比较好的解决办法是使用代码生成工具来生成这些代码。各MIS系统的三层架构的代码具有很大的相似性,这些代码依赖于数据库的架构信息,随着数据库架构信息的不一样有相应的变化。代码生成器以数据库架构信息为输入,输出三层架构的Model,BLL和DAL层的实现代码。代码生成器能够高效地生成源代码[3]。实践证明,使用代码生成器能够减少开发人员90%以上的编码量,缩短了项目开发周期,降低了开发成本。

收稿日期:2011-11-19;**修回日期**:2012-02-21
基金项目:贵阳市2010年重大科技专项项目([2010]筑科工合同字第6-01号)
作者简介:刘于新(1983-),男,贵州安顺人,硕士研究生,主要研究领域为可计算性与计算复杂性;姚凯学,教授,硕士生导师,主要研究领域为计算机控制技术、嵌入式系统设计;许道云,教授,博士生导师,主要研究领域为计算复杂性、可计算分析。

1 三层架构思想

在数据库系统的开发中,三层架构是最常用的分层架构设计。三层架构就是将系统的整个业务应用划分为表示层、业务逻辑层、数据访问层。采用分层结构主要是为了实现“高内聚、低耦合”[4]。

表示层(UI):位于三层架构的最外层,离用户最近。用于显示数据和接收用户输入的数据,为用户提供一种交互式操作的界面。将数据传递给业务逻辑层;同时也接收业务逻辑层传递的数据并以一定的格

式显示[5]。

业务逻辑层(BLL):主要处理来自表示层和数据访问层的数据,是表示层和数据访问层的纽带[6]。

数据访问层(DAL):其功能主要是负责数据库的访问操作。该层可以灵活地访问不同性质的数据库(如 SQL Server,Oracle 等),向业务逻辑层提供基于标准规范的通用数据格式数据[7]。

三层架构体现了面向对象的思想,程序编写人员将公用处理方法如数据库的增删查改写成公共方法,封装在类中供其他程序调用。不用每次操作数据库时都写那些重复的数据库操作代码[8]。

2 代码自动生成的设计与实现

2.1 代码生成器概述

三层架构的代码可以分成可变部分和不可变部分。可变部分的代码随着数据表的变化而发生变化,不可变部分的代码不会随着数据表的变化而变化。早期代码自动生成器把这两部分内容直接固化在程序中,如果要修改输出的目标代码,就必须修改代码自动生成器,程序可维护性差。现在大部分的代码自动生成器采用的都是基于模板的方法[9]。基于模板的方法就是根据代码结构编写模板,代码的不可变部分直接存储在模板中,可变部分使用标签将其嵌入在模板中。代码生成器对模板解释执行,就能输出得到目标代码。这种方法从模板中重用代码,提高了代码质量,避免了重复和枯燥的编码工作。基于模板的方法又分为两种类型:内置模板型和可定制模板型。

(1)内置模板型:代码的模板与生成工具是一体的,模板的格式和内容都隐含在生成工具中,不能定制。

(2)可定制模板型:代码的模板与生成工具分离,代码生成工具提供一种标记语言作为模板的编写规则。用户可通过标记语言编写代码模板。模板容易调整,维护。用户能够根据自己的需求修改模板来改变输出的目标代码,从而满足不同应用平台和应用环境的需要[10~12]。

基于模板的自动代码生成器一般有着必不可少的三个要素:第一是模板,也就是生成代码的格式和结构模板;第二是元数据,即在代码中需要建模的结构的相关资源;再就是业务规则,用于指定元数据与行为的规则[13]。

自动代码生成器生成的代码内容一般分为两种,一种是只生成代码框架,代码的实现部分需要程序人员手工编写;另一种是生成包括实现部分的所有代码。

文中设计的代码生成器采用基于. Net 平台的 C/S 架构,属于可定制模板型的代码生成器,生成的代码内容属于后一种类型的代码生成器。

要进行自动代码生成,需要确定元数据信息和如何根据三层架构的代码结构编写模板文件。

2.2 元数据

代码生成器的元数据包括两部分:数据库架构信息和相关 Xml 文件内容。Xml (Extensible Markup Language)是 W3C(World Wide Web Consortium)定义的可扩展标记语言。Xml 的可扩展性是指 Xml 允许用户按照 Xml 规则自定义标记。Xml 文件能够很好地体现数据的结构和含义[14]。

数据库架构信息包括的内容有:

数据库、表名称:当前选中的数据表信息,包括数据表和该表所在的数据库的名称。这些信息用来生成目标代码中的命名空间和类的名称。

字段信息:数据表中的字段的信息,包括字段名称、字段类型、字段长度、是否为空等。在生成 Model 和 DAL 层的代码时,要用到这些信息。如生成 Model 层代码时,需要遍历选中的表中的选中字段,为数据表中的选中字段生成相应的实体类的属性。

主键信息:数据表中主键的信息,包括主键名称、字段类型、字段长度等。

Xml 文件保存的内容有:

数据库中的字段类型和. Net 中的数据类型之间的对应关系。例如:数据库中 nvarchar 类型转换为 string 类型,money 类型转换为 float 类型。

C#数据类型和该数据类型缩写之间的对应关系。例如:int 对应 i,string 对应 str,bool 对应 bl;在生成代码时,在模板中添加上相应的标签,就能根据该变量类型在变量名称前加上该变量类型缩写。

数据库中字段的数据类型和 SqlDbType 枚举类型之间的对应关系。例如 nvarchar 对应 SqlDbType. NVarChar。

数据库中字段的数据类型和 C#有关数据类型转换函数之间的对应关系。例如数据库中字段的数据类型为 varchar,在. Net 环境中,使用 ToString 函数将 ExecuteReader 函数读出的数据库的该字段转换为字符串,即 varchar 对应 ToString 函数。

为了增加控制的灵活性和程序的可修改性,这些信息没有固化在程序中,而是以 Xml 文件格式保存在外部文件中,如果需要对转换的对应关系进行修改,只需要修改这些 Xml 文件,不需要改动程序。

与在目标代码中输出数据库架构信息不同,对于 Xml 文件保存的这部分元数据信息,在目标代码中输出的是转换之后的结果值。转换前,这些元数据信息都会被加载到解释引擎中,转换由解释引擎中的转换函数来完成。

2.3　模板设计

模板中的内容由三部分组成:静态对象、动态对象、模板脚本。

静态对象:是目标代码中不可变的部分,静态对象可以是任意的字符串,解释引擎将静态对象直接输出到目标代码中。

动态对象:用<% = % >标签表示动态对象,指示在标签的等号后输出一个值,该值可以是变量值或者函数的返回值。在生成. Net 三层架构的代码过程中,保存有数据库信息的变量的值和相关函数的返回值都会做为动态对象进行输出。

模板脚本:脚本语言使用 C#语言编写,使用. Net 环境提供的动态编译技术,模板的脚本语言能够处理复杂的逻辑。模板脚本由两部分内容组成。第一部分内容是代码块,用<% % >标签表示模板脚本。标签中的内容是模板的脚本语言。执行<% % >中的代码,结果输出到目标代码中。第二部分内容是方法,能在模板中定义方法并调用这些方法。定义方法开始标签<%% % >和结束标签<%# % >,标签的使用如下所示:<%% public string function(string para) { % >方法实现部分<%# } % >。在模板脚本的<% % >标签中调用方法,也可以使用<% = % >标签调用方法并输出返回值。

模板的编写可以使用常用的文本编辑器如记事本或者 word 来编写。编写完成后将文本保存为. tm 的模板文件。用户可以定制编写不同类型的模板,生成代码时,选择相应模板即可生成对应的代码。

2.4　代码生成过程

代码生成的流程如图 1 所示。代码生成过程分为三个阶段:加载引擎,解析模板,生成目标代码。加载引擎,代码解释引擎解释模板时,先要加载引擎,即将元数据读入引擎。解析模板,就是将模板内容分成一个个对象,根据对象的类型生成对应的 C#代码。将模板内容读入 CEngine 中,判断当前对象类型,读取当前对象,处理该对象。接着判断下一对象类型,读入并处理下一对象直到模板所有内容读取完成。模板解析结束,得到一个 C#类。生成目标代码,使用了. Net 提供的动态编译和反射技术。动态编译是指在程序运行时才生成 C#代码,然后在程序中使用 C#代码编译器动态编译这段 C#代码,最后反射调用它提供的功能。反射技术可以在运行时获得程序集中每一个类型的成员,还可以获得每个成员的名称、限定符和参数等。反射技术可以动态地创建类型的对象,即使这个对象的类型在编译时还不知道[15]。使用动态编译和反射使得模板脚本能够使用 C#的所有功能。生成目标代码时,将解析模板得到的 C#类动态编译生成临时程序集,再反射调用临时程序集中的自动生成代码函数,其返回值就是目标代码。

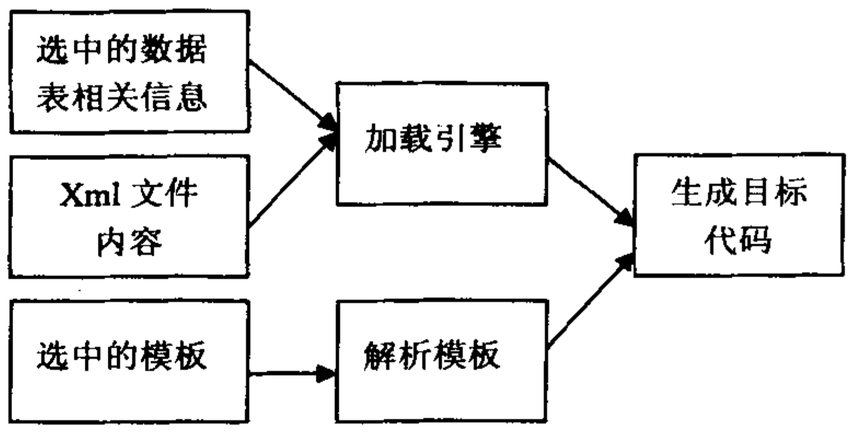

图 1　自动生成代码的流程

2.5　出错处理

用户编写的模板脚本难免会出现错误,例如模板开始标签和结束因为编写错误不能互相匹配,模板脚本的代码有语法错误,错误地引用了接口中不存在的元数据和函数等等。如果模板有错误,动态编译时,就会引发错误。这时,出现的错误提示信息有助于修改模板中的错误。为此定义 CErrors 类记录和管理这些错误提示信息。CErrors 含有三个属性。int ErrorLine 记录发生错误的行数,string ErrorMsg 记录错误信息,static bool IsHaveError 记录是否出现错误。动态编译结束,如果 IsHaveError 为真,则将 CErrors 的信息输出到 listBox,用户根据 listBox 中的提示信息,修改模板内容直到模板无误,正确生成目标代码。

3　应用举例

下面举例使用设计的自动代码生成器生成一段目标代码。在生成目标代码前,先准备好数据库和模板。使用 SQL Server 创建名为 DBSchool 的数据库,在数据库中添加一张数据表 tb_student。在表中添加字段 int stuID, varchar(50) stuName, bit stuSex,int stuAge, datetime stuBirthday。模板使用已经编写好的模板文件 DAL_TEMPLATE. tm。运行本程序,输入连接 SQL Server 的用户名,密码,进入程序界面。在左侧的树形控件中点击 DBSchool 数据库,选中 DBSchool 数据库中 tb_student 表。表中的所有字段的信息会在 dataGridView 控件中显示出来,在 dataGridView 控件中选择需要生成代码的字段。点击选择模板,在 openFileDialog 控件弹出的对话框中选择要使用的模板文件 DAL_TEMPLATE. tm,点击程序界面上的查看代码,就能看到解释引擎生成的代码。限于篇幅,下面列出的只是 DAL_TEMPLATE. tm 的查询函数的部分内容,出现省略号的地方表示省略的查询函数的内容。

public List<<% = codeGeneration. SelectedDbInfo. SelectedTableName % ><% = ModelClassNameSuffix % >>

Get<% = codeGeneration. SelectedDbInfo. SelectedTableName % >(<% = codeGeneration. SelectedDbInfo. SelectedTableName % ><% = ModelClassNameSuffix % > entity)

```
{
……
SqlParameter[ ] para = new SqlParameter[ ] { <%
            ColumnListIndex=0;
            foreach (Column c in codeGeneration. SelectedColumnList)
                {
                  if( ColumnListIndex= =0)
                      {%>new SqlParameter( "@ <%=c. ColumnName%>" , <%=codeGeneration. ConvertDBTypeToParameter( c. ColumnDBType)%>)<%
                        ColumnListIndex++;
                  }
                else
                      {%>, new SqlParameter( "@ <%=c. ColumnName%>" , <%=codeGeneration. ConvertDBTypeToParameter( c. ColumnDBType)%>)<%
                }%><%
            }%>;
……
}
```

模板中的 foreach 表示的是循环语句的开始。模板脚本中的 codeGeneration. SelectedColumnList 表示 dataGridView 选中的数据表的字段，codeGeneration. ConvertDBTypeToParameter（c. ColumnDBType）表示将数据库数据类型转换为 SqlDbType 枚举类型的函数，函数中的参数表示的是字段的数据类型。ColumnName 表示的是字段名称。下面列出了生成的对应的目标代码。

```
public List<tb_studentInfo> Gettb_student ( tb_studentInfo entity)
        {
……
SqlParameter[ ] para = new SqlParameter[ ] { new SqlParameter( "@ stuID" , SqlDbType. Int) , new SqlParameter ( "@ stuName" , SqlDbType. VarChar) , new SqlParameter ( "@ stuSex" , SqlDbType. Bit) , new SqlParameter( "@ stuAge" , SqlDbType. Int) , new SqlParameter( "@ stuBirthday" , SqlDbType. DateTime) };
……
}
```

4 结束语

文中设计的自动代码生成器目前只支持 SQL Server 数据库。只需要将绑定 dataGridView 的函数扩展到能够从 Access，Oracle 等数据库中读取数据库的信息，就能将支持的数据库扩展到 Access，Oracle 等数据库。

自动代码生成器给实际的数据库系统开发带来极大的好处：代码生成器生成的代码风格统一，很利于系统维护和团队协作；代码生成器将开发人员从枯燥繁重的代码编写工作中解放了出来，提高了他们的工作效率，使他们能将精力集中在数据库系统的分析与设计上，提高了项目的质量；代码生成器生成的代码具有很强的健壮性，添加到解决方案里面就能直接使用。代码生成器具有的这些优点使得它在实际项目开发中有着广泛的使用。

参考文献：

[1] 赵跃华，王 凌. 基于敏捷方式的 Java 代码生成方法的设计[J]. 计算机工程与设计，2009，30(12)：3018-3021.

[2] Jörges S，Margaria T，Steffen B. Assuring property conformance of code generators via model checking[J]. Formal Aspects of Computing，2011，23(5)：589-606.

[3] Schaefer J，Stynes J，Kroeger R. Model-based Performance Instrumentation of Distributed Applications [DB/OL]. 2008 [2011 - 11]. http://www. springerlink. com/content/b641x484174w4104/.

[4] 胡 炜. 第三方物流管理系统的设计与实现[J]. 价值工程，2010，29(13)：30-31.

[5] 夏克付. 基于三层架构的企业用户服务系统设计[J]. 电脑知识与技术，2010，6(27)：7470-7471.

[6] 张 荣，王培俊，曹永彦，等. 基于 ASP. NET 技术的实验中心信息化管理平台的设计[J]. 计算机技术与发展，2011，21(5)：59-60.

[7] 于家潭，邵宝民，黄宝香，等. 基于. net 2.0 三层架构的青岛市数字城建档案馆[J]. 计算机技术与发展，2010，20(7)：235-237.

[8] 钟红春. ASP. NET 2.0 程序设计教程[M]. 北京：人民邮电出版社，2009.

[9] Rumpe B. Agile Modellierung mit UML：Codegenerierung，Testfälle，Refactoring[M]. New York：Springer-Verlag，2005.

[10] 冉春娟，黄 华. 基于关系数据模型代码生成器的设计与实现[J]. 湖北大学学报(自然科学版)，2010，32(2)：151-156.

[11] Solberg A，Oldevik J，Aagedal J ∅. A Framework for QoS-Aware Model Transformation，Using a Pattern-Based Approach [DB/OL]. 2004[2011-11]. http://www. springerlink. com/content/90yadr4f86xw1p5w/.

[12] Bork M，Geiger L，Schneider C，et al. Towards Roundtrip Engineering - A Template-Based Reverse Engineering Approach [DB/OL]. 2008[2011-11]. http://www. springerlink. com/content/j06l3187357276g5/.

[13] 苗维杰，李天辉. 基于 XML 代码生成技术的应用研究[J]. 电子元器件应用，2009，11(10)：75-78.

[14] 耿祥义. XML 基础教程[M]. 北京：清华大学出版社，2006.

[15] 朱有产，李玉凯，李自强. 基于. NET 反射技术的规约插件实现原理[J]. 继电器，2006，34(22)：60-63.

一种基于改进的非对称裁剪中值滤波算法

作者： 陆钢锋， 袁保社
作者单位： 新疆大学信息工程与技术学院,新疆乌鲁木齐830046
刊名： 计算机技术与发展
英文刊名： Computer Technology and Development
年，卷(期)： 2012(8)

参考文献(15条)

1. 赵跃华;王凌 基于敏捷方式的Java代码生成方法的设计[期刊论文]-计算机工程与设计 2009(12)

2. J(o)rges S;Margaria T;Steffen B Assuring property conformance of code generators via model checking 2011(05)

3. Schaefer J;Stynes J;Kroeger R Model-based Performance In strumentation of Distributed Applications 2008

4. 胡炜 第三方物流管理系统的设计与实现[期刊论文]-价值工程 2010(13)

5. 夏克付 基于三层架构的企业用户服务系统设计[期刊论文]-电脑知识与技术 2010(27)

6. 张荣;王培俊;曹永彦 基于ASP.NET技术的实验中心信息化管理平台的设计[期刊论文]-计算机技术与发展 2011(05)

7. 于家潭;邵宝民;黄宝香 基于.net 2.0三层架构的青岛市数字城建档案馆[期刊论文]-计算机技术与发展 2010(07)

8. 钟红春 ASP.NET 2.0程序设计教程 2009

9. Rumpe B Agile Modellierung mit UML:Codegenerierung,Tesff(a)lle,Refactoring 2005

10. 冉春娟;黄华 基于关系数据模型代码生成器的设计与实现[期刊论文]-湖北大学学报(自然科学版) 2010(02)

11. Solberg A;Oldevik J;Aagedal J Φ A Framework for QoS-A-ware Model Transformation,Using a Pattern-Based Approach 2004

12. Bork M;Geiger L;Schneider C Towards Roundtrip Engineering-A Template-Based Reverse Engineering Approach 2008

13. 苗维杰;李天辉 基于XML代码生成技术的应用研究[期刊论文]-电子元器件应用 2009(10)

14. 耿祥义 XML基础教程 2006

15. 朱有产;李玉凯;李自强 基于.NET反射技术的规约插件实现原理[期刊论文]-继电器 2006(22)