

基于 B/S 三层架构的 ASP.NET 系统的代码生成研究

杨学增,王满敬

(西南交通大学 信息科学与技术学院,四川 成都 610031)

摘要:根据 B/S 信息系统的结构和功能特点,通过对代码生成工具的工作模式的研究,提出了基于 MDA 模型驱动架构和基于模板的驱动模式相结合的代码生成模式。它包括三种基本的模型:功能模型、对象模型、交互模型。此工具通过这三种基本模型和模板来实现 B/S 三层构架信息系统的生成,非常适合在迭代开发的项目中使用。

关键词:代码生成;ASP.NET;三层结构

中图分类号:TP311.11

文献标识码:A

文章编号:1672-7800(2009)07-0011-03

0 引言

随着信息化的普及,企业对软件的需求量越来越大,软件工程师面临的工作任务也越来越多。在开发过程中,开发人员经常面对重复性编码,重复性模块设计。为了提高开发效率,提高软件复用程度,使软件开发人员从重复、繁琐的工作中解放出来,更能关注于客户的需求,代码生成工具应运而生。

目前,代码生成工具的工作模式主要有基于 MDA(Model Driven Architecture)模型驱动架构和基于模板的驱动模式。MDA 主要的思想是把 PIM(Platform Independent Model)平台无关模型转化成 PSM(Platform Specific Model)平台相关模型。MDA 虽然有很多的标准规范来支持,但在实际的生成中,由于体系庞杂以及缺乏必要的模型信息,生成的代码并不理想。基于模板驱动的工作模式:首先用户按照模板的规则设计模板,然后模板引擎对模板进行读取和解析,生成目标文件。该模型生成的代码容易定制,但没有充分利用分析和设计阶段的模型信息,自动化生成程度不高。本文通过对企业信息系统的研究以及结合这两种模型,提出了基于 B/S 三层构架的 ASP.NET 系统的代码生成模型。

1 模型结构及工作原理

由于企业信息系统的特性,本文代码生成是较高层次生成代码,从三种基本的模型到代码实现。由于需求要和设计及实现保持一致性,所以这三种模型有很多的附加信息(涉及到 UI、存储、业务流程等信息),而且在生成时,提供了中间的设计和实现平台的配置选项,用户可根据需要来选择相应的设计框架和平台。结合 UML 分析、设计阶段的模型,本系统定义了三种基本模型:功能模型、对象模型、交互模型。

1.1 功能模型

本功能模型是一种类似 UML 中的用例模型,如图 1 所示。

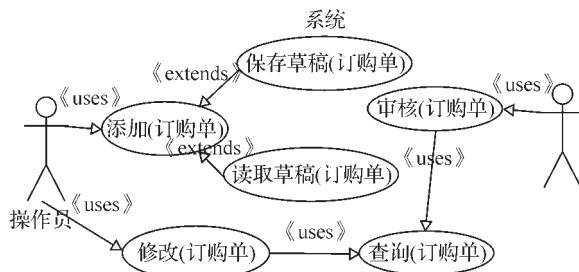


图 1 功能模型

在功能模型中,信息系统中的角色与 UML 用例模型中的参与者相对应,功能(服务)用动宾词组表示。本模型中动词表示系统提供的服务功能,宾语代表领域模型中的对象。在功能的粒度上,本模型采用了一个角色可以独立操作完成的一个事务作为它的功能。在功能的关系上,模型依然采用了扩展关系和包含(使用)关系。

1.2 对象模型

对象模型是对 UML 中的领域模型和类或对象的扩展,它包括 E-R 模型信息,服务信息以及 UI 显示信息。也就是说对象模型包含了三层(表示层、业务逻辑层、持久化层)信息。对象模型必须和面向对象的类以及 E-R 模型中的实体建立联系,而且它们的关系是一致的。

根据对功能模型进一步的分析,得到了图 2 的对象模型图。

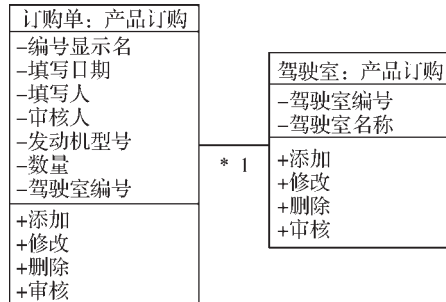


图 2 对象模型

在这个模型中,订购单表示了领域模型中的对象,产品订购表示订购单所在的包或者它的命名空间。其中订单中包括其属性信息和服务信息。对象模型中属性信息既包括了基本的属性,也包括 UI 的特点和对象(实体)之间的关联关系(为了数据库中操作的方便,这种关系可能在数据库中是虚拟的)。对象中的服务功能是由功能模型中所有角色对该对象的使用确定的。对象模型中的属性信息主要包含了以下信息:基本信息(属性名称、数据类型、数据长度、精度、默认值、数据格式正则表达式),UI 信息(对象的显示名称、是否显示、属性的顺序、属性的显示名称、控件表示),数据存储信息(是否为空、主键、外键、有无虚拟主键、是否为虚拟外键、是否为虚拟主键、组成主键的数目、外键的数目、组成虚拟主键的数目)。

1.3 交互模型

在 B/S 模式下,用户与系统的交互都是通过一个个功能页完成的。每个功能页面要和功能模型的一个或有关联的一组功能相对应,它提供用户访问接口。被包含用例有两种使用方式,第一种提供一个功能页面和包含用例的功能页面的调用接口,第二种作为包含用例功能的一个调用。扩展用例和被扩展用例共用一个功能页面。每个功能模块都要为用户提供操作结果信息。由于功能和业务流程都比较固定,根据功能特点,划分了几个功能页,其中每个页面包含若干个功能。系统提供的主要功能页可以分为添加编辑页、草稿页、查询页、查看页、审核页、自定义页等。其中添加编辑页是复用页面,即可以添加编辑功能,根据功能模型中的扩展用例,添加模型还应该包含保存草稿的功能。根据上面的功能模型和业务流程,设计了它的交互模型,如图 3 所示。

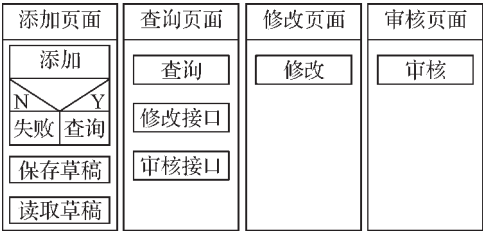


图 3 交互模型

每个功能操作都对应着成功或失败。例如添加页面中的添加功能,它执行完后有两种结果,若成功则跳转到查询页面,若失败则弹出失败对话框(系统定义),确认失败后,回到原页面。其中接口只是该功能页面执行被包含用例后提供的用户访问接口。接口可以在它所对应的功能页中执行,也可以直接调用。例如查询页面中的查询功能是一个被包含用例,根据被包含用例的使用规则,它提供一个功能页和包含用例的接口。菜单为每一个业务模块组织相应的功能页面,而且每个功能页都对应它的角色权限,每个角色都有相对应的菜单。业务流程通过各个功能页面接口的调用顺序来实现。所以系统的业务流程通过菜单中功能页面的组织形式和每个功能页的调用关系来表示。

1.4 对应关系

通过以上的研究,得出模型与代码映射的关系,如表 1 所示。

表 1 模型与代码的对应关系

模型	代码
对象模型中的属性和关系	Model 层中的实体类 数据库中的表
对象模型中的对象功能	业务逻辑层的服务
功能模型中的功能	系统接口和角色权限
功能模型中的参与者	角色
交互模型中的功能页面	UI 层的用户接口
交互功能模型中的接口和调用	功能流程

2 设计与实现

2.1 元数据的定义

由功能模型、对象模型和交互模型分别定义表示这 3 种模型的元数据。描述功能模型的元数据,采用了 xml 的文件格式。下面是对象模型中订购单的元模型示例。

```
<objects>
  <namespaces>
    <namespace id="1" description=" 产品订购" name="ProductOrder"/>
  </namespaces>
  <object id="1" name="OrderBill" displayName="订购单">
<namespace id="1"></namespace>
    <property name="CabCode">
      <displayName>驾驶室编号</displayName>
      <ishidden>否</ishidden>
      <type>string</type>
      <length>36</length>
      <isfk>否</isfk>
      .....
      <ref>
        <parent id="2" property="CabCode" displayName="驾驶室编号" hidden="是"></parent>
        <display>
          <parent id="2" property="CabName" displayName="驾驶室名称"></parent>
        </display>
      </ref>
      <dataformats>
        <dataformat description="驾驶室标号只能由数字和字符组成的 36" regex="\w{36}">
      </dataformat>
      </dataformats>
    </property>
    .....
  </object>
</objects>
```

2.2 模板的定义和模板解析器

根据实际的需要,本文没有采用第 3 方的模板引擎,而是设计了自己的模板规则和相应的解析器。

模板的主要内容包括:注释、内置关键字、内置控制结构(顺序、判断、循环)、宏定义、内置常量、变量以及用户的文件内容。为了与用户文件的内容相区分,规定所有的语法格式都是 [OUT X],要求 [OUT X]的内容全部大写。其中 X 可以为变量和关键字。模板的语法规则举例:

宏定义:它的格式为 [OUT DEFINE X],X 为宏开关,表示已经定义了 X。取消 X 的定义 [OUT UNDEFINE X]即可;

Repeat 循环的结构:

```
[OUT REPEAT]
private [OUT BUILTIN TYPE] [OUT BUILTIN FIELD-NAME]
[OUT ENDREPEAT]
```

本文利用上述语法规则定义了 3 种内置功能页的模板和控件选择模板。控件模板的的格式如下:

```
[OUT LABEL]
<asp:Label ID="lbl[OUT PROPERTYNAME]" runat="server"></asp:Label>
[OUT END]
[OUT TEXTBOX]
<asp:TextBox ID="tb [OUT PROPERTYNAME]" runat="server" MaxLength ="[OUT FIELDLENGTH]"></asp:TextBox>
[OUT END]
```

2.3 配置选择

工程配置文件包括了项目名称、项目位置、各层的结构名称、存放路径、模板名称、界面方式、数据访问方式、自定义开关等配置,如图 4 所示。

2.4 代码生成器

Generator 主要读取模板信息,根据模型的元数据和内置标识符,调用模板解析器进行相应的解析生成。在三层的结构中,分别有 6 个(ASPXGenerator、ASPXCSTGenerator)和 1 个 WebControlsGenerator 与显示层相对应,BLLGenerator 与业务逻辑层相对应,DALGenerator 与数据持久层相对应,ModelGenerator 与实体类相对应。主要完成以下转换:对象模型到数据库的映射、存储,对象模型到实体类的映射,功能模型在业务逻辑中的映射,



图 4 项目模板配置

交互模型和对象模型的属性信息在 UI 中的映射(参看表 1)。

3 结束语

本文通过对代码生成工具的工作模式的研究,提出了一种基于模型和模板的代码生成模型。该模型对 3 种基本的模型进行了修改和扩展,建立了模型与代码之间的映射关系。该模型可以用于生成基于 B/S 三层构架的 asp.net 信息系统。目前系统的通用性和界面的交互性还有一定的局限性,需要进一步的提高。

参考文献:

[1] Martin Flower.UML 精华(第 3 版)[M].徐家富,译.北京:清华大学出版社,2007.

[2] Anneke Kleppe,Jos Warner,Wim Bast.解析 MDA[M].鲍志云,译.北京:人民邮电出版社,2004.

[3] 万建成,孙彬.支持用户界面自动生成的界面模型[J].计算机工程与应用,2003(18).

[4] 张静,孔芳,杨季文.一个基于 Java 的代码生成工具的设计与实现[J].电子学与计算机,2007(6).

(责任编辑:杜能钢)

Research of Code Generation in ASP.NET System Based on B/S Three-Layer Architecture

Abstract:According to the characteristics of B/S information system structure and function and studying the work mode of code generation tools, a research model which combines MDA-based and template-based code generation models is proposed in this paper. It includes three basic models: functional model, object model, interactive model. This tool, through the three basic models and templates to generate information system of B/S three-layer, is fit for the project with iterative development.

Key Words:Code Generation;ASP.NET;Three-layer