Software Engineering and Applications 软件工程与应用, 2013, 2, 98-103
http://dx.doi.org/10.12677/sea.2013.25018 Published Online October 2013 (http://www.hanspub.org/journal/sea.html)

Study on Software Development Methods Driven by Metadata*

Xiaohang Li[1#], Xiaopeng Hu[1], Zhengqing Han[2]

[1]School of Information Science and Technology, Southwest Jiaotong University, Chengdu
[2]School of Electrical Engineering, Southwest Jiaotong University, Chengdu
Email: #xhli_scce@home.swjtu.edu.cn

Received: Aug. 15th, 2013; revised: Aug. 23rd, 2013; accepted: Sep. 10th, 2013

Abstract: The management of entity objects with large number of properties and mass data exchange with external data source are common requirements in lots of application software development, while its implementations are very similar. A metadata-based unified implementation scheme is proposed to mitigate the duplicate code and to improve productivity of the programmer. The scheme addresses an object-oriented design paradigm, and a general mass data exchange framework with Microsoft Excel as mediator. The proposed methods improve the abstraction level of software design and the adaptability to changing requirements, and practical applications validate the feasibility of the methods.

Keywords: Metadata Model; Object-Oriented Design Paradigm; Mass Data Exchange

基于元数据的软件开发方法研究*

李晓航[1#]，胡晓鹏[1]，韩正庆[2]

[1]西南交通大学，信息科学与技术学院，成都
[2]西南交通大学，电气工程学院，成都
Email: #xhli_scce@home.swjtu.edu.cn

收稿日期：2013 年 8 月 15 日；修回日期：2013 年 8 月 23 日；录用日期：2013 年 9 月 10 日

摘　要：针对应用软件开发中有大量属性的实体对象的管理、以及与外部数据源批量数据交换等需求的普遍性和实现的相似性，本文提出了基于元数据的统一实现方法，包括一种以元数据为基础的面向对象设计方法，以及以 Excel 为中介的通用数据批量交换处理框架。该方法提高了软件设计的抽象化水平和适应需求变化的能力，实际系统验证了方法的可行性。

关键词：元数据模型；面向对象设计；数据批量交换

1. 引言

在开发不同的计算机软件系统的过程中，经常要实现一些相似性很高的需求，其中典型的有：1) 采用面向对象技术开发以关系数据库作为数据持久化方案的软件系统时，针对有大量属性的实体对象的赋值、保存和显示的需求；2) 开发基于数据库的软件系统时，数据库和外部数据源(如 Excel 文件、带分隔符的文本文件)的批量数据交换(包括批量增加、修改和导出)的需求。

针对第一个需求，现有的开发方式是在完成业务领域建模后，在程序代码中将类的属性以“硬编码”的方式固化下来，然后通过引入对象/关系映射(O/R

*基金项目：铁道部重点课题(2011J023-C)。
#通讯作者。

Mapping, ORM)层，将类的属性映射为数据库中表的字段，并通过 ORM 层的驱动，完成对象的持久化。可见，现有的开发方式需要在三个地方重复定义类的属性，即类的程序代码、ORM 映射文件和数据库表的字段，如图 1 所示。

在系统发布后如果由于需求变化要增加类的属性(或修改现有属性的约束条件)，则对 ORM 映射文件和数据库字段，只需修改相关配置；而对以“硬编码”方式实现的类的代码，则必须要手工修改，然后重新编译并部署系统，这使得系统很难适应需求的频繁变化，增加了后期系统维护的代价。此外，对管理信息系统(MIS)类应用而言，还有一种常见的情况，即 MIS 中的实体类的大多数属性通常不会参与业务逻辑计算，它们的作用仅仅是存储某项信息，在需要的时候能够显示在用户界面中即可。例如在笔者开发的某人事管理系统中，人员类有 40 多个属性，但只有人员状态、所在部门、人员级别、参加工作时间等几个属性需要参与业务逻辑计算(如计算人员工资)，而其它大多数属性(如身份证号、出生日期、住址、联系方式、民族、籍贯等)往往只需要在用户界面中能显示和保存即可。可见，对于 MIS 而言，采用“硬编码”在程序代码中实现实体类的所有属性，必要性不大。

针对第二个需求，目前通常有两种解决方法：1) 利用 Bulk Insert 之类的 T-SQL 语句；2) 开发专用的数据导入程序[1]。方法 1：一般只能由了解数据库系统结构的技术人员完成，并且交互性较差，一般只能作为建库时的一次性操作，无法实现程序界面的交互式录入功能；方法 2：引入了配置信息和开发工具内部的元数据，但仍需要根据不同的数据源开发专用的转换程序，并非一个框架层面的通用解决方案。

为了更好地解决软件开发中的这类共性需求，可以采用以应用层元数据为核心的开发方法，通过元数据来解耦需求变化和具体实现。众所周知，元数据是“关于数据的数据”[2]，它在数据库管理系统(DBMS)和程序开发环境(如 MS Visual Studio、Java)中，都得到了广泛的应用。但 DBMS 或开发环境提供的元数据，缺乏构建应用程序所需的一些关键信息[3]，而应用层元数据可以提供更加丰富的数据关联信息、类型信息和展示信息，并且独立于应用系统和开发环境，更加有利于构建具有较高的抽象化水平，且能适应不同应用需求的框架程序。例如，文献[4]通过应用层元数据配置(Custom Metadata Configuration)提出了基于元数据的模式语言(Pattern Language)，并为基于元数据的框架的内部结构提出了参考结构；文献[5]基于应用层元数据，设计了一个快速软件开发平台；文献[3]提出了一种基于数据库表的元数据模型，该模型既能描述表之间的一般关系(例如参照关系)，又能描述表之间基于面向对象的继承关系，同时实现了数据操纵 SQL 语句和数据维护程序界面的自动生成。在文献[3]提出的元数据模型的基础上，本文提出了基于动态属性的面向对象设计方法，以及以 Excel 为中介的通用数据批量交换处理框架，提高了软件设计的抽象化水平和适应需求变化的能力。

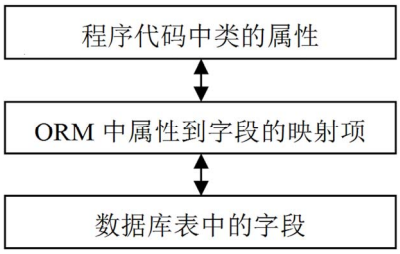

Figure 1. Position of class attributes in OOD
图 1. 面向对象设计中类的属性出现的位置

2. 数据库表的元数据模型

文献[3]提出的以数据库表为中心的元数据模型的类图如图 2 所示，其中 TTableBase 类是元数据模型中表的基类，它由 TField (字段)类组成。TTableBase 有两个派生类，一个是 TTable (具体表)类，对应于一个实际的数据库表；另一个是 TVirtualTable (虚表)类，用来描述表与表之间基于面向对象语义的继承关系，可以支持任意层次的继承。TTableBase 的 GetSelectDataSql 方法用来构造数据查询所需的 SQL 语句，而 GetInsertDataSql 、GetUpdateDataSql 和 GetDeleteDataSql 方法分别用来构造插入、修改和删除数据记录所需的 SQL 语句；TField 类的 ValueValid 方法对将要赋给 TField 对象的 CurrentValue 属性的字符串的合法性进行判断，如数据类型是否正确、是否不允许为空、是否符合正则表达式格式等。

3. 基于元数据的面向对象设计方法

利用文献[3]中的元数据模型，设计了由元数据驱

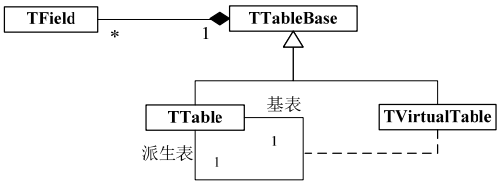

Figure 2. Class diagram of database table's metadata model
图 2. 数据库表元数据模型类图

动的、带有动态属性和对象/关系映射能力的实体类 TEntityByMetaData，如图 3 所示。TEntityByMetaData 是基于元数据的面向对象设计方法中业务类的基类。

3.1. TEntityByMetadata 的动态属性架构

为了避免在程序代码中以“硬编码”的方式实现类的属性，TEntityByMetadata 支持动态属性架构，该架构通过以下三个方面来实现。

首先，TEntityByMetadata 与 TTableBase 构成 1:1 的关联关系，意味着在运行时，TEntityByMetadata 对象的所有动态属性来源于对应的 TTableBase 对象的字段列表(即 TField 对象集合)，其属性名称为 TField.Name、属性类型为 TField.Type，属性的有效性约束(如是否允许为空、最大最小值、正则表达式等)和 TField 对于字段的各种约束相同，TTableBase 提供了 TEntityByMetadata 的属性定义。

其次，为了存储动态属性的值，在 TEntityByMetadata 中定义了一个字符串数组(用 Values 表示)，该数组的项目数等于相关的 TTableBase 对象拥有的 TField 对象的数量，而每个属性值在数组中的下标，和相关的 TField 对象在 TTableBase 的字段列表中的下标一致。

最后，为了对动态属性的值进行统一存取，在 TEntityByMetadata 中设计了属性 PropValueByName，这是一个通过属性名称对属性值进行存取的接口，并将属性值统一用字符串类型表示。PropValueByName 的实现包括读取和写入两个方面，其流程如图 4 所示。需要注意的是在写入流程中，在将输入的属性值(字符串类型)实际赋给 Values 数组项目之前，要分别根据 TField 的数据类型和完整性约束条件对输入串的格式进行检查，以确保在统一的字符串类型下，TEntityByMetadata 中存储的动态属性的值始终满足相关的 TField 对象的类型和有效性要求。

采用动态属性后，如果由于需求变化导致类的属

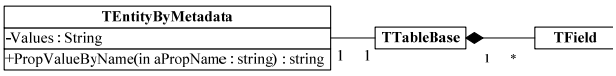

Figure 3. Entity class based on metadata class
图 3. 利用元数据类构造的实体类

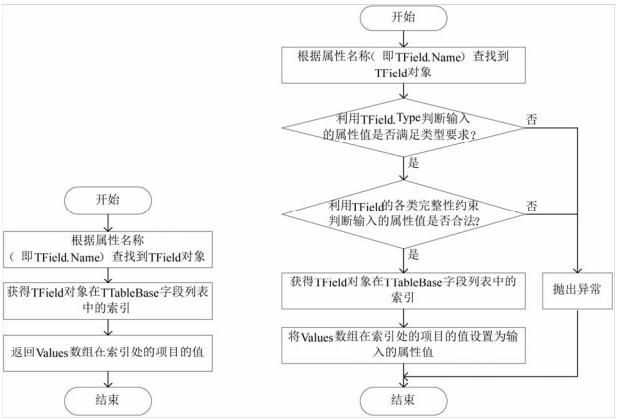

Figure 4. The reading and writing flow chart of PropValueByName; (a) Reading flow chart; (b) Writing flow chart
图 4. PropValueByName 的读取和写入流程图；(a) 读取流程图；(b) 写入流程图

性发生变化，则只需要修改数据库表结构和元数据定义，而无需对程序代码进行修改、重新编译和部署。

3.2. TEntityByMetadata 的对象/关系映射

在这一结构中，利用以 TTableBase 为基础的元数据模型，可以实现从 TEntityByMetadata 对象到数据库记录之间的对象/关系映射，即对象的持久化功能。对象/关系映射的流程图如图 5 所示。

3.3. 模型应用

在实际应用中，对于一个有持久化需求的具体业务类层次，应按照如下步骤完成建模：

1) 按照文献[3]的要求在关系数据库中建立类层次的关系模型，即为每个类创建一个数据库表，并建立起基表和派生表之间的 1:1 的参照关系。

2) 创建类层次的元数据模型，在元数据模型中每个 TTable 对象对应一个业务类，每个 TVirtualTable 对象表示业务类层次中一个从根节点到叶节点的继承关系。元数据模型一般在程序启动时一次性创建，并作为单件(Singleton)对外提供服务。

3) 在程序代码中以 TEntityByMetadata 为基类，实现业务类层次，TEntityByMetadata 起到了关联业务类层次和元数据模型的作用。在实例化业务类层次中

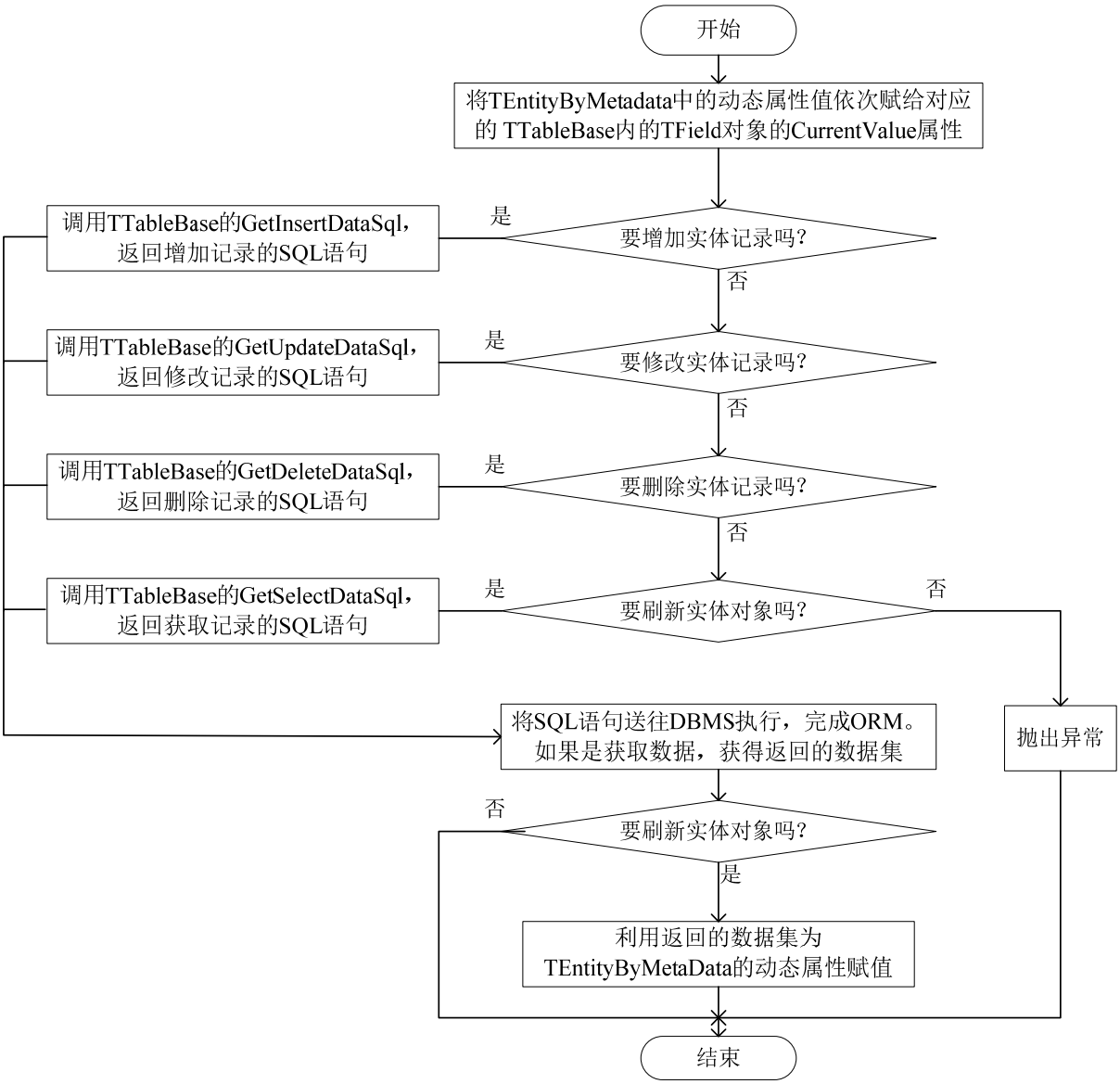

Figure 5. O/R mapping flow chart of TEntityByMetadata object
图 5. TEntityByMetaData 对象的对象/关系映射流程图

的每个具体类(Concrete Class)时，会关联业务对象对应的 TTableBase 对象，并保存进 TEntityByMetadata 中针对 TTableBase 的成员变量中。如果具体类是业务类层次的根，则关联的是 TTable 对象；否则关联的是 TVirtualTable 对象。

基于 TEntityByMetadata 实现的业务类层次的示意图如图 6 所示，其中四个业务类分别对应四个 TTable 对象(记为 Table1、Table2、Table3 和 Table4)，两个业务类继承关系(业务类 2 到业务类 3、业务类 2 到业务类 4)分别对应两个 TVirtualTable 对象(记为 VirtualTable1 和 VirtualTable2)。在实例化具体类时，

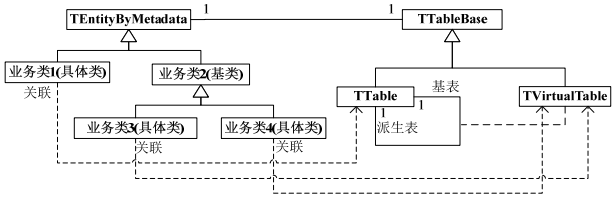

Figure 6. Business class hierarchy based on TEntityByMetadata
图 6. 基于 TEntityByMetadata 实现的业务类层次示意图

业务类 1 的实例将关联 Table1 对象、业务类 3 和业务类 4 的实例分别关联 VirtualTable1 对象和 VirtualTable2 对象。

默认情况下，业务类的所有属性都是动态属性，

可以根据元数据的变化自动调整，并通过 PropValueByName 接口进行存取。但如果业务类的某些属性需要参与业务逻辑计算，则仍然可以像传统方式一样创建固定属性，不过固定属性的实际值仍然保存在 TEntityByMetadata 的 Values 数组中，并在内部通过 PropValueByName 进行存储。例如，假设人员类(TPerson)的属性“出生日期”(BirthDate)需要参与业务逻辑计算，则仍然应该为 TPerson 创建 BirthDate 属性，其类型为日期型(如 Java 中的 Date 型)，属性的读取和写入要通过将日期－字符串转换函数作用于 PropValueByName['BirthDate']来实现。

通过动态属性和固定属性相结合的方式，即保证了与业务逻辑关系密切的属性的明确性，又实现了与业务逻辑无关的属性的动态性，能自适应由于需求变更引起的元数据的变化。此外，只需对文献[3]中的通用数据维护框架稍加改进，就能自动生成对 TEntityByMetadata 派生类对象的维护(浏览、增加、修改)界面，从而减少程序界面的开发工作量。

4. 基于元数据的数据批量交换处理框架

以文献[3]的元数据模型为基础，设计了与具体应用需求无关的数据批量交换(包括批量增加、修改和导出)处理框架。在数据批量交换时，通常采用 Excel 作为前端数据处理工具。由于 Excel 本身具有强大的数据编辑和公式计算能力，并且用户对 Excel 的操作一般比较熟悉，所以用 Excel 有助于用户对大批量数据的快速编辑，并能显著减少程序内部专用数据录入界面的开发工作量。

以 Excel 为前端工具，基于元数据的数据批量交换的基本流程如图 7 所示。

首先要利用选中对象的元数据(即 TTableBase 对象)自动创建录入数据的 Excel 模版，该创建过程可以用开发工具中 Excel 的 OLE 接口实现[6]。为了使模版可以脱离系统独立使用，设计了一种自包含的模版格式，包括报头区和数据区。报头区是只读的，由模版的前三行组成，第一行包括模版对应的 TTableBase 对象的 Name 和 PRCName 属性(一般对应表的中文名)，第二、三行是要录入的字段的标识，即 TField 对象的 Name 和 PRCName 属性。Name 的值用于在导入时反向查找 TTableBase 和 TField 对象，PRCName 的值用于用户对目标对象和字段的识别。模版数据区的每一行是一条独立的记录。对于批量增加，数据区默认为空，由用户输入；对于批量修改，模版的字段列表中会自动包含主键字段，而数据区由需要修改的现有数据记录组成，这些数据记录通常通过程序内的查询功能返回。另外，如果一个字段是参照字段(即 IsReference 属性等于 True 的字段)，则要求在模版中录入数据时，录入字段的参照值，将来由导入程序将其转换为编码，这样有助于用户的理解。典型的录入数据的 Excel 模版见表 1。

在 Excel 模版中直接录入数据后，利用 Excel 的 OLE 接口，可以将数据读入程序内存，然后按以下步骤进行处理：

1) 根据报头中保存的 TTableBase.Name，查找对应的 TTableBase 对象；

2) 对于数据区中输入的每一条记录，完成如下工作：

a) 对于记录的每个字段，根据报头中保存的字段标识(即 TField.Name)，查找对应的 TField 对象；

b) 利用 TField.ValueValid 对字段单元格中输入的字符串的合法性进行校验，如果有任何不合法数据，则不再导入该记录，直接转到下一条记录；

c) 将每个单元格(cell)的值赋给对应的 TField 对象(即设置 TField.CurrentValue 属性)，如果该字段是参

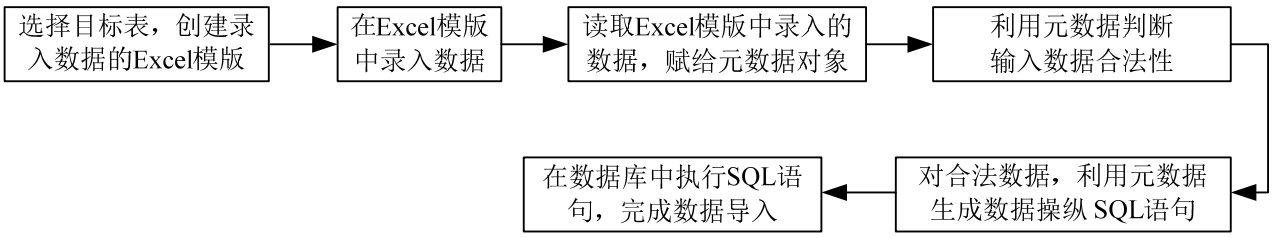

Figure 7. Process of general mass data exchange based on metadata
图 7. 基于元数据的数据批量交换的基本流程

Table 1. The excel template for human information input
表 1. 用于人员信息批量录入的 Excel 模版

1) 批量增加人员模版(The template for bulk inserting)			
tb_Person		人员信息	
Name	Sex	Identity No.	Birth Date
姓名	性别	身份证号	出生日期

2) 批量修改出生日期模版(The template for bulk updating)		
tb_Person	人员信息	
PK_ID	Name	Birth Date
人员主键	姓名	出生日期
610B2069-8787-40C8-9056-A0547BA279A6	张三	1995-12-2
5B477903-2FE8-421E-BAA9-3E322CA20300	李四	1994-1-15

照字段，则赋值时会自动进行参照值到编码的转换；

d) 对于模版中不存在而 TTableBase 拥有的其它 TField 对象，如果 TField 的默认值(DefaultValue)属性不为空，则把默认值赋给 TField，如果 TField 是自动生成值的字段(如 GUID 类型字段)，则自动产生一个新值(如一个 GUID 字符串)赋给 TField；

e) 利用 TTableBase 的 GetInsertDataSql (对于批量增加)或 GetUpdateDataSql (对于批量修改)获得该记录对应的数据操纵 SQL 语句；

f) 将该 SQL 语句送往数据库引擎执行，完成当前记录在数据库中的增加或修改。

5. 结论

以一种应用层元数据模型为基础，提出了基于动态属性的面向对象设计方法，以及以 Excel 为中介的通用数据批量交换处理框架。在笔者参与开发的铁道部课题和多项 MIS 系统中应用该方法，显著减少了开发工作量，提高了系统适应需求变化的能力。

6. 致谢

本文的主要工作是在铁道部重点课题——2011J023-C 的研究和开发过程中完成的，特此表示感谢。

参考文献 (References)

[1] M. Fowler. Using metadata. IEEE of Software, 2002, 19(6): 13-17.
[2] J. D. Ullman, J. Widom. A first course in database systems. Prentice Hall, 1998: 7-8.
[3] 李晓航, 胡晓鹏. 基于元数据的通用数据维护框架设计[J]. 计算机工程, 2010, 36(20): 80-82.
[4] E. Guerra, F. Alves, U. Kulesza, et al. A reference architecture for organizing the internal structure of metadata-based Frameworks. The Journal of Systems and Software, 2013, 86: 1239-1256.
[5] 蔡昭权, 卢庆武, 郑宗晖. 基于元数据的快速开发平台设计与实现[J]. 计算机工程, 2009, 35(9): 60-62.
[6] 金勇, 兰放. 基于同步模式的 EXCEL 与.NET 数据交互[J]. 武汉理工大学学报－信息与管理工程版, 2010, 32(4): 557-560.

