基于 C#快速生成 word 报告①

肖 斌[1]，李 超[1]，汪 敏[2]

[1](西南石油大学 计算机科学学院，成都 610500)
[2](西南石油大学 电气信息工程学院，成都 610500)

摘 要：在进行信息系统的开发过程中，在 word 文档中特定位置填入从数据库中读取的数据资料来自动生成 word 文档报告，已成为办公自动化系统的重要组成部分。分析了传统 word 对象模型生成海量表格数据速度慢的弊端，结合微软 Office Word2003 对 XML 的支持特性，通过 Word 文档和 XML 文档的相互转换，提出利用 StringTemplate 将 DataTable 的数据转换成特定格式的 xml 字符串，实现海量表格数据的快速导出。

关键词：Word 模板；StringTemplate；XML

Generating Rapidly Word Report Based on C#

XIAO Bin[1], LI CAO[1], WANG Min[2]

[1](School of Computer Science, Southwest Petroleum University, Chengdu 610500, China)
[2](Schoo of Electrical Engineering and Information, Southwest Petroleum University, Chengdu 610500, China)

Abstract：When developing an information system, filling in data information at the specific position of word documents to automatically generate documents' reports has played an important part in Office Automation System. This paper analyzes the traditional Word object model, which has disadvantages of generating mass of table data slowly. In conjunction with the good supporting characteristics of Microsoft Office Word 2003 for XML, and by way of the interconversion of Word document and XML document, this paper proposes that converting Data table into XML of specified format by utilization of StringTemplate can realize the fast derivation of mass table Data.

Key words：Word template; StringTemplate; XML

1 引言

Word 不但具有无与伦比的图文处理能力，同时也提供了 Word 层次结构的对象，使用户可以用 VBA 或者第三方软件开发工具实现文档自动化。因此，越来越多的软件系统开始采用 Word 作为报表输出工具，有的甚至将 Word 作为数据录入工具[1,3]。但是 C#与 Word 交互时，每次都需要调用 I/O 接口，在处理海量表格数据时，如果循环对每个单元格赋值，那是非常耗资源的，导出速度慢。

本文提出另外一种办法来解决这个问题。利用 ADO.NET 取得数据集，取出数据集中的 DataTable 数据，将数据转换为泛型集合 Dictionary 以树型节点的方式进行存储。利用 office2003 支持 xml 的特性并结合 StringTemplate 的视图与业务分离的特性将动态的业务数据最终以 xml 格式的方式写入文档内容。提高生成 word 报告的速度。

2 传统的生成word报表实现原理

将 Word 作为报表输出或者录入工具，一般都是采用制作 Word 模板的方法来解决字符的定位。WORD 的模板文件其实就是通过书签来添加内容的。也就是通过在 WORD 文档中创建书签，然后在程序中获取模板文件的所有书签，通过给书签赋值来进行文档生成[2]。在 C#中，需要调用 OFFICE 的 WORD 组件通过 C#来操作 WORD，进而生成 WORD 文档。Word 层次结构的对象如图 1 所示：

① 收稿时间:2011-11-03;收到修改稿时间:2011-12-05

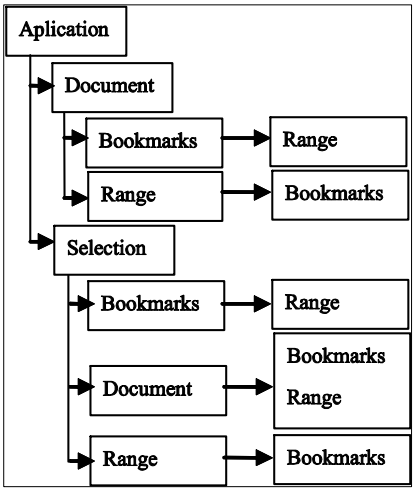

图 1 Word 层次结构的对象

程序的操作流程如下：?声明 WORD 程序的对象 → 声明一个 WORD 文档对象 → 获取当前的操作文档对象 → 获取文档所有的书签 → 将数据库数据赋值到对应的书签 → 将文档另存为指定的文件夹下。在生成指定位置文档内容时采用代码如下：

```
//循环所有的书签，并给书签赋值
    for (int oIndex = 0; oIndex < testTableremarks.Length; oIndex++)
    {
      obDD_Name = WD + testTableremarks[oIndex];
      doc.Bookmarks.get_Item(ref
      obDD_Name).Range.Text=p_TestReportTable.Rows[0][testTablues [oIndex]].ToString();
    }
```

通过上述代码可以看出，生成 word 报告时，采用的是逐个单元格的存取方式，操作速度慢，效率低，实验表明，在对动态表格进行数据填写时，如果读取的表格数据量大，将导致程序导出 WORD 报表的速度相当慢，效率及其低下，原因如下：

1. 频繁的进行 IO 操作比较耗资源。

2. Word 操作调用的是 com 接口，跨进程操作影响操作速度。

3. 基于 xml 技术的 WORD 报表系统设计

基于传统的 VBA 或 com 技术生成 word 报表中海量数据数度慢的现状，本文提出利用 office 2003 支持 xml 的特性，结合 StringTemplate 模版视图与业务分离的特性，将 DataTable 的数据转换成 XML 格式的字符串，利用 File.WriteAllText(FileName, s)完成在文件中插入指定格式的字符串，一次 IO 操作完成文件的写操作，提高生成 Word 报告的效率。系统模型图如下所示：

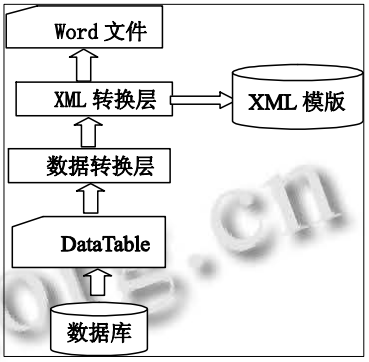

图 2 海量数据导出 WORD 模型图

本系统中，通过 ADO.NE 获取系统数据，并将数据存于 DataTalbe 中，在数据转换层，按照 xml 模版中的后台数据格式进行转换，XML 转换层利用事先设计好的 XML 模版，将数据库数据动态的以 XML 格式进行呈现。

3.1 基于 WordprocessingML 的 XML 模版设计

微软从 Office Word 2003 开始已经针对 XML 进行了完整设计，使其支持称为 Word 标记语言(WordML)的原生 XML 词汇。它描述了如何将一份 Word 2003 文档以及相关联的部分（如字形、字体、表格）以 XML 文档的形式表现。通过对 WordprocessingML 编程来操作 Word，可以在不用引入第三方库的情况下，把 Word 当作 XML 文本来操作，并且可以在没有安装 Word 的机器上运行[4]。最简单的 WordProcessingML 文档仅仅包含五种基本元素和一个命名空间（namespace），它们是：

(1) wordDocument 元素——WordProcessingML 文档的根元素；

(2) body 元素——WordProcessingML 文档中可显示文本的容器；

(3) p 元素——段落标记；

(4) r 元素——相邻的带有一系列属性集的 WordProcessingML 元素的集合；

(5) t 元素——文本的一个片断；

Template 文件：

```
<?xml version="1.0"?>
```

```
<w:wordDocument
xmlns:w="http://schemas.microsoft.com/office/word
/2003/wordml" >
<w:body>
<w:tbl>
<w:tr>
  $ItemToDisplay.Properties.keys:{k |
<w:tc>
<w:p>
<w:r>
<w:rPr>
<w:b w:val="on"/>
<w:t>
  $ItemToDisplay.Properties.(k);format="XML"$
</w:t>
</w:rPr>
</w:r>
</w:p>
</w:tc>
}$
</w:tr>
$ItemToDisplay.SubNodes.keys:{k
|$ItemToDisplay.SubNodes.(k):{y |
$word_item(ItemToDisplayRec= y)$ }$}$
</w:tbl>
  </w:body>
</w:wordDocument>
```

在该模版文件中，视图要展现的动态数据来源于 ItemToDisplay，这是本系统中视图和业务解耦的设计体现，数据的业务处理由 StringTemplate 来提供 ItemToDisplay。从而达到以模板来产生动态的 xml 字符串。

3.2 基于 StringTemplate 的设计与实现

StringTemplate（简称 ST）是一个基于 Java 的模板引擎库，支持 C#。StringTemplate 的显著特点是严格执行模型视图分离，使用 StringTemplate 可以严格保证业务逻辑和表现逻辑相分离，不会互相干扰，提高网站的开发和维护的效率。模型和视图分离所带来的优点包括：模板文件可以在相似的网站开发中重用，清晰的模板文件可以作为网站开发的说明文档，模板文件可以单独修改[5]。为了给 XML 提供数据源，后台实现代码如下：

```
StringTemplate stNeed = stgCollection.
GetInstanceOf(Template);
stNeed.SetAttribute("ItemToDisplay", strTemplete);
string s=stNeed.ToString();
```

利用该后台代码，实现业务数据和模版的关联，从而将 DataTable 的数据以字符串的形式来表现。通过 File.WriteAllText(FileName, s)，一次性的将字符串数据存储于 word 文件中，提高了海量数据的存储速度。

3.3 数据转换层的设计与实现

Dictionary 泛型集合是 C# 泛型集合之一，与之对应的非泛型集合为 HashTable 类，使用泛型集合类可以提供更高的类型安全性和更高的性能，避免了非泛型集合重复的装箱和拆箱。本系统采用采用泛型集合的形式来存储转换数据。首先定义抽象类 public abstract class HashTableAddRecursive<T> : Dictionary <string, T>，然后定义类 public class HashTable Add RecursiveProperties : HashTableAddRecursive<object> 以用来存储节点数据。通过 HashTableAdd Recursive Properties 产生对象包含数据集合，每个数据项包含键和值两部分。这些集合以树节点的方式添加到树的节点中，为方便程序的扩展，设计了接口 INode,在该接口中声明了节点的级数、名字和唯一标识等字段项，声明了添加对象的方法，声明集合属性 HashTableAdd RecursiveProperties Properties，并利用该属性的 add 方法来存储 DataTable 的数据集合。Inode 的实现由 public class ST_Node<T> : INode 来实现。接口设计如下：

```
public interface INode
{
int iLevel { get; set; }
string strName { get;set;}
string UniqueID { get; set; }
Type oType { get; }
void AddProperty(string Key);
HashTableAddRecursiveObjects AddObjects();
AddObjects();
void AddObject();
HashTableAddRecursiveProperties Properties
{ get; }
HashTableAddRecursiveCollection SubNodes
{ get; }
```

```
}
```

创建节点对象 strTemplete，并作为树节点的根节点，该节点对象的 Properties 属性包含 DataTable 的字段名等信息。实现如下：

```
foreach (ExportProps exp in properties)
{
  strTemplete.Properties.Add(exp.NameProp, exp.DisplayNameProp);
}
```

其中，List<ExportProps> properties 存储的是数据表字段的集合。

上述代码实现了字段信息在 strTemplete 节点上的存储，而行记录数据由 strTemplete 节点对象的子节点提供。创建节点对象 dtRow，节点对象的 properties 属性为一个泛型集合对象，利用 dtRow 对象的 properties 属性包含每行记录的所有字段值，在集合项中，key 为字段名，value 为字段值，dtRow 即为一行数据的集合。然后将存储一行数据的 dtRow 对象添加到 HashTableAddRecursiveObjects objRows 对象中，这样，objRows 将包含所有行记录的转换。objRows 即为行记录的泛型集合。部分代码如下：

```
HashTableAddRecursiveObjects objRows = new HashTableAddRecursiveObjects();
int i = 0;
foreach (DataRow dr in dt.Rows)
{
INode dtRow = new ST_Node<DataRow>(dr, null, null);
foreach (ExportProps exp in properties)
{
object o = dr[exp.NameProp];
if (o != null && o != DBNull.Value)
{
string value = null;
value = o.ToString();
dtRow.Properties.Add(exp.NameProp, value);
}
Else
dtRow.Properties.Add(exp.NameProp, "");
}
objRows.Add("K" + i, dtRow);
i++;
}
```

通过数据转换模块的实现，完成了从数据表行记录到泛型集合的转换，从而获得数据节点的信息，通过 strTemplete.SubNodes.Add("A", objRows);完成子节点的添加。从而为 xml 模版提供标准格式的数据源。

4 总结

利用 word 作为应用程序的报告输出工具已成为办公自动化系统的重要组成部分，利用 word 提供的编程接口技术可以高效的生成各种格式的 word 报告，但在生成海量数据时，由于频繁的操作 I/0 接口，导致速度极其缓慢。本文提出借助 StringTemplate 的视图和业务分离的特性，将 DataTable 数据对象以泛型集合的方式存储，最终转换成 xml 格式字符串的方式操作 word 报告，从而使导出速度得到数量级的提高。同时该解决方案也为 DataTable 导出为 excel,pdf 等格式文件提供了一种新思路。

参考文献

1 鲁保玉,杨新芳.用 Delphi 生成 Word 报告及动态结构表格.计算机软件与应用,2007,24(3):180−183.
2 孔令彦,董蓬勃,姜青香,等.使用 Visual Basic 操作 Microsoft Word 对象生成报告文档.计算机工程与应用,2003,39(36):115−117.
3 范明虎.OLE 和 Word 对象模型在题库管理系统开发中的应用.计算机工程与设计,2007,28(10):2487−2490.
4 开放的 XML 开发官方网站.http://openxmldeveloper.org/default.aspx.
5 官方网站 www.stringtemplate.org.