Из опыта автоматизации Word на языке C# на примере создания оглавления

А. Н. Вильданов

Нефтекамский филиал
Башкирский государственный университет
452681, Нефтекамск, ул. Трактовая, 1

e-mail: *alvild@mail.ru*

Аннотация. Описан алгоритм создания содержания документа в формате MS Word на языке C# в среде Microsoft Visual Studio 2010 Express (на примере сборника трудов конференции). Программно встраиваемое содержание имеет возможность автоматического обновления нумерации страниц. Исследованы возможности C# по форматированию текста вордовского документа, сохранению в формат PDF и т. п. Приведены основные команды по работе с документом Word на языке C#. Разработанное в статье приложение показывает, что с помощью C# можно достаточно эффективно решать широкий круг задач, связанных с автоматизацией рутинных действий с документами Microsoft Word. Рассмотренные в статье приемы и методы могут оказаться полезными при разработке подобных десктопных приложений.
Ключевые слова: Microsoft Word, COM, C#, Microsoft Visual Studio 2010 Express.

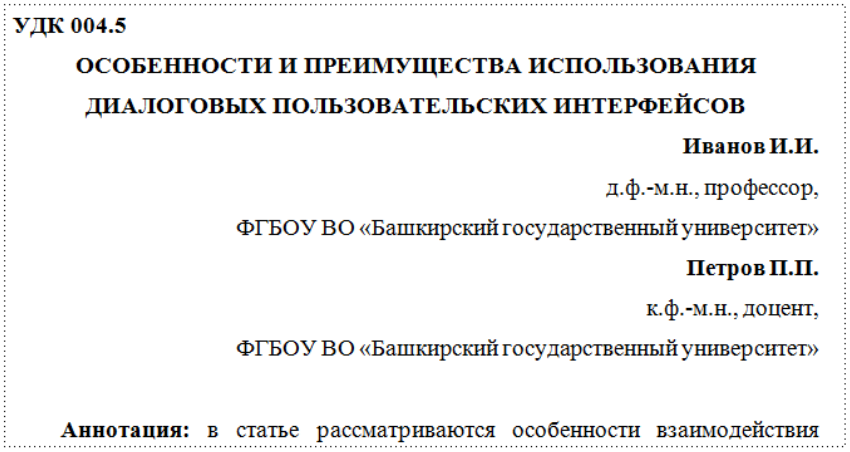
1. Постановка задачи

Как известно, Microsoft Word является COM-объектом [1], т. е. спроектирован таким образом, что позволяет другим программам подключаться к себе и управлять им. Программно можно проделать практически все операции, которые мы делаем вручную в Word: создать новый документ, внести в него правки, сохранить его и т. п. Для такой автоматизации действий можно, например, подключиться к MS Word с помощью таких языков программирования высокого уровня, как Delphi, C++, C# и т. д. Можно, конечно, и просто создать макрос прямо внутри документа на языке Visual Basic for Applications. Как делать — дело вкуса, привычки и квалификации программиста.

В данной работе выбор пал на язык C#, который позволяет достаточно быстро создать прототип приложения для Windows. Имеется в виду, что язык C# сразу (без дополнительных библиотек) содержит готовые классы и методы, отвечающие за решения многих современных актуальных задач для десктопных приложений в ОС семейства Windows.

В качестве среды программирования в работе выбрана среда Microsoft Visual Studio 2010 Express. Несмотря на год выпуска, набор средств для создания приложений в нем весьма внушителен, имеется большой плюс — бесплатность.

Одной из распространенных задач при работе с документами является создание содержания. Понятно, что для достаточно больших документов ручная работа по его созданию становится тяжелой. Word имеет достаточно много средств и настроек для создания автоматического содержания (например, в Microsoft Word 2010 это подменю «Ссылки» → «Оглавление»). Но в некоторых случаях этот способ не подходит. Представим себе, например, сборник трудов конференции. Тогда заголовок и авторы могут находиться на разных строках (рис. 1):



УДК 004.5

ОСОБЕННОСТИ И ПРЕИМУЩЕСТВА ИСПОЛЬЗОВАНИЯ ДИАЛОГОВЫХ ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ

Иванов И.И.
д.ф.-м.н., профессор,
ФГБОУ ВО «Башкирский государственный университет»

Петров П.П.
к.ф.-м.н., доцент,
ФГБОУ ВО «Башкирский государственный университет»

Аннотация: в статье рассматриваются особенности взаимодействия

Рисунок 1. Пример заголовка статьи конференции

2. Предлагаемый алгоритм решения

Как можно решить задачу составления содержания в таком случае? Желательно, чтобы нумерация страниц в содержании могла автоматически обновляться при внесении изменений в документ. В данной статье предлагается следующий способ. Нужно разметить в документе те элементы, которые мы собираемся разместить в содержании, определенными стилями. После пройтись по всему документу и сохранить в специальном массиве все встреченные элементы с созданными стилями.

Например, создадим и отформатируем стиль для заголовков статьи. Дадим ему, например, название “header_article”. Для упрощения программирования стили будем присваивать не отдельным словам и предложениям, а целым абзацам. При нашем подходе он не обязательно должен иметь стиль заголовка, а может быть создан на основе стиля обычного абзаца.

Для авторов придумаем стиль “author”. В сборник конференций в содержание могут входить и названия секций. Поэтому предусмотрим стиль “section_name”. Таким образом пробежимся по всему документу и обработаем все статьи (рис. 2.).

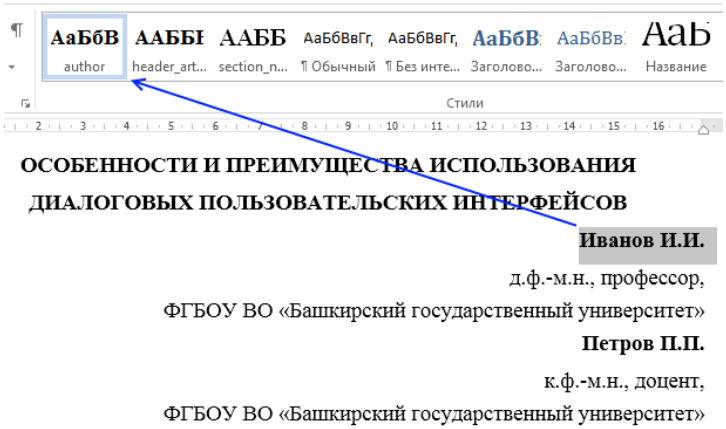


Рисунок 2. Присвоение стилей

Каким теперь будет алгоритм для составления содержания? Сначала нужно организовать цикл по всем абзацам документа, в котором:

- считываются текст и стиль абзаца;
- для абзацев с заголовком статьи, со стилем “header_article”, создаются закладки с названием вида “bookMarkParagN”, где N — номер абзаца (можно, конечно, называть и по-другому);
- авторы статьи и название (заголовок) статьи сохраняются в массиве SelParagraphs.

Обежав весь документ, мы получим массив с данными о названиях всех статей и об их авторах. По этому массиву уже можно построить содержание, например, в виде таблицы из двух столбцов. В первом столбце размещаются авторы статьи и название статьи. Во втором столбце создаются перекрестные ссылки на номера страниц закладок.

3. Программная реализация

Итак, создадим в среде Microsoft Visual Studio 2010 Express новое приложение Windows Forms. Чтобы в нашем приложении были доступны классы Microsoft Word, нужно добавить ссылку на соответствующую библиотеку. Выбираем в меню «Проект» → «Добавить ссылку», открываем вкладку COM, выбираем Microsoft

Word 14.0 Object Library (рис. 3). Цифра зависит от версии установленного MS Word.

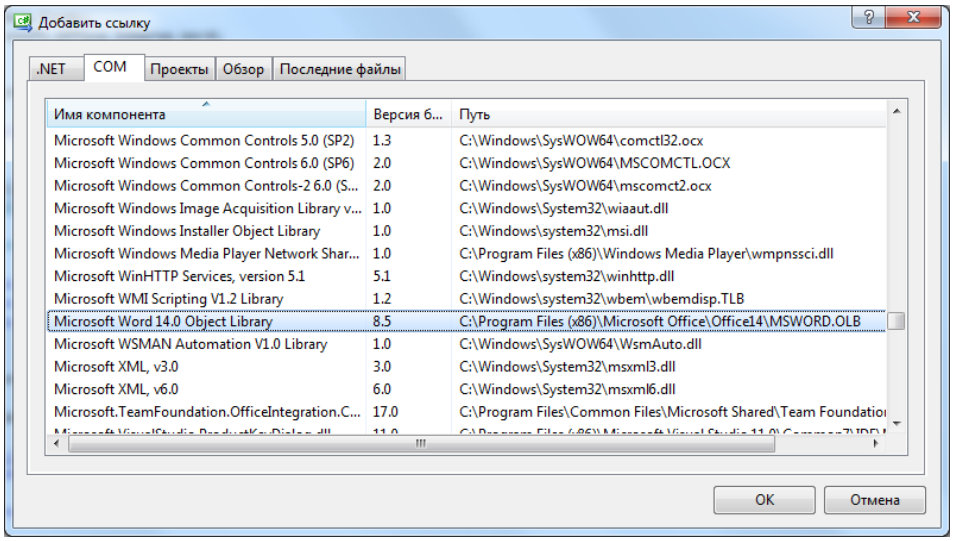


Рисунок 3. Подключение библиотеки для работы с MS Word

Теперь нам доступны методы пространства имен Microsoft.Office.Interop.Word**.** Для упрощения обращений к нему создадим псевдоним для него:

```
using Word = Microsoft.Office.Interop.Word;
```

Нам понадобятся две глобальные переменные для работы с Word [2]:

```
Word._Application app;
Word._Document doc;
```

Первая переменная представляет, собственно, само приложение MS Word, а вторая — наш вордовский документ.

Создадим класс contents для хранения информации об элементах будущего содержания:

```
class contents
{
public string authors="";
public string article_name="";
public string bookMark="";
}
```

Пункты authors и article_name хранят информацию об авторах статьи и название статьи. Поле bookMark будет хранить название закладки на заголовок статьи. Чтобы в полученном содержании иметь возможность автоматического обновления нумерации страниц, нужно будет создать перекрестные ссылки на созданные закладки.

Итак, сначала пробежимся по всем абзацам текста. Многие основные текстовые единицы в Word представляют собой коллекции [3] (объекты, напоминающие массивы, со встроенными методами и свойствами). Например, абзацы представляют собой коллекцию Paragraphs, закладки — коллекцию Bookmarks и т. д. [4]. Нужно иметь в виду, что нумерация в коллекциях начинается с единицы (в отличие, например, от массивов в Java, где первый элемент имеет номер ноль). Запускаем цикл:

```
for (int i = 1; i < doc.Paragraphs.Count; i++) {
```

Ключевым классом для работы с содержимым документа является Range. Range можно понимать для себя просто как фрагмент документа, но, конечно, представляющий собой объект со своими свойствами и методами [5]. Например, с помощью свойства Text мы получим текстовое содержимое абзаца:

```
string WordP = doc.Paragraphs[i].Range.Text;
```

Мы ищем абзацы определенного стиля, поэтому нам понадобится метод get_Style() для чтения стиля абзаца:

```
string WordS =
 ((Word.Style)doc.Paragraphs[i].get_Style()).NameLocal;
```

С помощью get_Style() мы можем определить, когда попался заголовок статьи, создать закладку и сохранить в массиве данные статьи:

```
if (WordS == "header_article")
{
String bookmarkName = "bookMarkParag" + i;
app.ActiveDocument.Bookmarks.Add(bookmarkName,
              doc.Paragraphs[i].Range);
contents c1 = new contents();
c1.article_name = WordP.Trim();
c1.bookMark = bookmarkName;
SelParagraphs.Add(c1);
}
```

Аналогичным образом поступаем с названиями секций. Пробежав весь документ, приступим к созданию таблицы:

```
Word.Table myTable = doc.Tables.Add(rng, 1, 2);
В цикле создаем строки таблицы:
for (int i = 0; i < SelParagraphs.Count; i++)
{
    myTable.Rows.Add();
```

Обращаемся к элементам массива для получения данных об авторах, названии статьи, названии закладки:

```
String authors = SelParagraphs[i].authors;
String article_name = SelParagraphs[i].article_name;
String bookMark = SelParagraphs[i].bookMark;
```

В первый столбец размещаем авторов и название статьи:

```
myTable.Rows[i+1].Cells[1].Range.Text =
           authors + " " + article_name;
```

Теперь во второй столбец нужно вставить перекрестную ссылку на страницу заголовка статьи. Для этого имеется метод InsertCrossReference:

```
cellRange = myTable.Rows[i + 1].Cells[2].Range;
object InsertAsHyperlink = true;
cellRange.InsertCrossReference(
Word.WdReferenceType.wdRefTypeBookmark,
Word.WdReferenceKind.wdPageNumber,
bookMark,
ref InsertAsHyperlink
);
```

Четыре параметра InsertCrossReference соответственно означают, что:

1) тип ссылки — перекрестная ссылка;

2) сведения, включаемые в перекрестную ссылку — это номер страницы;

3) далее идет название закладки;

4) является ли перекрестная ссылка гиперссылкой на соответствующий элемент — истина.

При таком добавлении наблюдается небольшой глюк — перекрестная ссылка почему-то добавляется не во второй столбец, а в начало первого столбца. Этот глюк лечится сворачиванием диапазона ячейки или уменьшением границы диапазона [6]:

```
cellRange = myTable.Rows[i + 1].Cells[2].Range;
cellRange.MoveEnd(Word.WdUnits.wdCharacter, -1);
```

Теперь гиперссылка попадает во второй столбец. При нажатии на номер страницы можно перейти на статью (рис. 4):

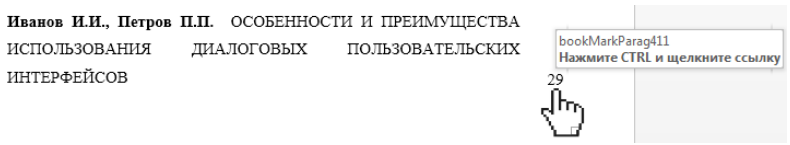


Рисунок 4. Переход к статье по ссылке из содержания

В принципе, название статьи мы также могли оформить в виде гиперссылки, но это чревато проблемами: при обновлении содержания или сохранении в формате PDF слетает форматирование, и содержание документа принимает недооформленный вид.

Потом можно выделить и отформатировать первый столбец таблицы:

```
myTable.Columns[1].Select();
app.Selection.ParagraphFormat.Alignment =
     Word.WdParagraphAlignment.wdAlignParagraphJustify;
app.Selection.ParagraphFormat.FirstLineIndent = 0;
app.Selection.Font.Name = "Times New Roman";
app.Selection.Font.Size = 14;
```

Названия методов выглядят громоздкими, но понятными. Сохраняем изменения в документе, закрываем документ, закрываем Word:

```
app.ActiveDocument.Save();
doc.Close();
app.Quit();
```

4. Сохранение в PDF-формат

После публикации сборника часто требуется перевести весь сборник или отдельные страницы (постатейно) в PDF-формат. В C# есть готовый метод перевода документа целиком:

```
doc.SaveAs( "D:\\док.pdf",
           FileFormat: Word.WdSaveFormat.wdFormatPDF );
```

Чтобы перевести не все, а только указанные страницы в PDF-формат, есть метод ExportAsFixedFormat. Для сохранения всех статей сборника в PDF номера страниц можно взять из второго столбца нашего содержания:

```
int from = Convert.ToInt32(
     myTable.Rows[i].Cells[2].Range.Text
);
int to =  Convert.ToInt32(
     myTable.Rows[i+1].Cells[2].Range.Text
) - 1;
```

Название файла можно позаимствовать из первого столбца (авторы и название статьи).

```
String articleName = myTable.Rows[i].Cells[1].Range.Text;
String pdfFileName = "D:\\Файлы\\" +
              " " + articleName + ".pdf";
```

В цикле создаем PDF-файлы:

```
doc.ExportAsFixedFormat(
pdfFileName,
Word.WdExportFormat.wdExportFormatPDF,
false,
Word.WdExportOptimizeFor.wdExportOptimizeForPrint,
Word.WdExportRange.wdExportFromTo,
from,
to
);
```

Третий параметр, равный false, означает, что после создания pdf-файл не нужно открывать. Последние два параметра отвечают за диапазон страниц.

5. Заключение

Разработанное приложение доказывает, что с помощью C# можно достаточно эффективно решать широкий круг задач, связанных с автоматизацией рутинных действий с документами в Microsoft Word. Но есть и существенный недостаток — низкая скорость работы технологии COM. Для обработки сборника объемом около ста страниц уходит до пяти минут.

Существуют два решения по увеличению скорости обработки документов. Первый — отказаться от прямого перебора всех абзацев. Вместо этого искать нужные фрагменты соответствующих стилей с помощью встроенного в `Range` объекта `Find`. Правда, алгоритм при этом будет более сложным и запутанным.

Второй, кардинальный способ — отказаться от технологии `COM`, а работать непосредственно с файлом `.docx` как с архивом из файлов формата `Office Open XML` с помощью прямых методов доступа к текстовым файлам [7]. Скорость при этом будет значительно выше. Правда, нужно хорошо разбираться в этом формате, который является достаточно сложным.

Литература

[1] *Вильданов А. Н.* Разработка приложений в Delphi : учеб. пособие. — Уфа : РИЦ БашГУ, 2012.

[2] Профессиональная работа с текстом Word Expert. [Электронный ресурс]. URL: http://wordexpert.ru/forum/viewtopic.php?id=2611

[3] *Гильманов Р. Ф.* Автоматизированное создание договоров из шаблонов программным путем на основе объектной модели word документа // *Академическая публицистика*. 2017. № 5. С. 15–20.

[4] Общие сведения об объектной модели Word. Microsoft.com. [Электронный ресурс]. URL: https://msdn.microsoft.com/ru-ru/library/kw65a0we.aspx

[5] *Корняков В. Н.* Программирование документов и приложений MS Office в Delphi. — СПб. : БХВ-Петербург, 2005.

[6] Writing programmatically lines of text in a Word cell. Stack Exchange Inc. [Электронный ресурс]. URL: https://stackoverflow.com/ques-tions/15166728/writing-programmatically-lines-of-text-in-a-word-cell

[7] *Елманова Н.* Коротко об OpenXML // *КомпьютерПресс*. 2007. № 7. С. 158–160.

***Автор*:**
Алмаз Нафкатович Вильданов — кандидат физико-математических наук, доцент кафедры математического моделирования и информационной безопасности, Нефтекамский филиал ФГБОУ ВО «Башкирский государственный университет»

From experience of Word automation in C# language on the example of creation of contents

A. N. Vildanov

Neftekamsk branch of Bashkir State University
Traktovaya st., 1, Neftekamsk, Russia, 452681

e-mail: *alvild@mail.ru*

*Abstract***.** Describes the algorithm for creating the contents of a document in MS Word format in C# language in Microsoft Visual Studio 2010 Express environment (using the collection of conference proceedings as an example). Programmatically embeddable content has the ability to automatically update page numbering. The possibilities of C# for formatting the text of the Word document, saving in PDF format, etc. are explored. The main commands for working with the Word document in C# are given. The application developed in the article shows that using C# can solve quite effectively a wide range of tasks related to automation of routine actions with Microsoft Word documents. The techniques and methods discussed in this article can be useful in the development of such desktop applications..
Key words: Microsoft Word, COM, C#, Microsoft Visual Studio 2010 Express.

References

[1] *Vildanov A. N.* (2012) Razrabotka prilozheniy v Delphi : uchebnoye posobiye. Ufa, RIZ BSU, 96 p. [In Rus].

[2] http://wordexpert.ru/forum/viewtopic.php?id=2611 [In Rus].

[3] *Gil'manov R. F.* (2017) *Nauchnyj ehlektronnyj zhurnal «Akademicheskaya publicistika»,* 5:15–20. [In Rus].

[4] https://msdn.micro-soft.com/library/kw65a0we.aspx

[5] *Kornyakov V. N.* (2005) Programmirovanie dokumentov i prilozhenij MS Office v Delphi. SPb, BHV-Peterburg. [In Rus].

[6] https://stackoverflow.com/questions/15166728/writing-programmatically-lines-of-text-in-a-word-cell

[7] *Elmanova N.* (2007) *Komp'yuterPress*, 7:158–160. [In Rus].