Convertendo arquivos no formato doc (Word 97) para docx (Word 2007)

# Introdução

A alteração mais visível no Office 2007 foi, sem dúvida, a introdução da nova interface, com o uso da faixa de opções (Ribbon), que facilitou muito o uso do programa, deixando as funções muito mais acessíveis ao usuário. Uma outra mudança, mais escondida, está no novo formato de arquivos.

Este formato é baseado no OpenXML (<http://openxmldeveloper.org/default.aspx>) e tem uma série de vantagens:

* É um formato aberto e padronizado. Ao contrário dos formatos antigos, que eram proprietários e não documentados, o novo formato é completamente documentado e sua padronização foi aprovada pela ECMA.
* Sua estrutura interna é conhecida, e pode ser manipulada por programas desenvolvidos por qualquer um. Um arquivo Office 2007 é um arquivo zip com outra extensão (docx, xlsx ou pptx) contendo pastas e arquivos xml comuns. A Microsoft introduziu um novo namespace (System.IO.Packaging) no .net 3.0, que permite manipular os arquivos e criar novos, usando qualquer linguagem compatível.
* Em geral, são menores que os arquivos doc. Como são arquivos zip contendo dados xml, têm boa compactação e têm tamanhos bem menores que seus equivalentes do Office 97.
* Podem ser abertos com o Office 2000, 2003 e XP, através do Office Compatibility Pack, que pode ser baixado gratuitamente em <http://office.microsoft.com/en-us/products/HA101686761033.aspx>

Você pode converter seus arquivos de texto antigos no formato novo abrindo-os no Word e escolhendo a opção “Salvar Como”, usando o formato “docx”, ou a opção “Converter” do Word 2007, porém este é um processo longo e tedioso, especialmente se você tiver muitos arquivos em seu disco rígido. Este artigo irá mostrar como fazer um programa em C# que converte os arquivos selecionados, em um determinado diretório e em seus subdiretórios.

# Abrindo arquivos do Office

Quando queremos abrir um arquivo do Office em nosso programa, podemos criar funções para acesso aos seus dados internos. Esta é uma tarefa complicada, sujeita a erros e passível de modificação entre as versões, não sendo recomendada.

Outra maneira de abrir estes arquivos é usar o VSTO, um add-in para o Visual Studio, que está disponível para dowload em <http://msdn2.microsoft.com/en-us/office/aa905533.aspx>. Porém esta versão pode ser instalada apenas nas versões Professional e Team System do Visual Studio (ou comprada a parte, para qualquer versão do VS), o que inviabiliza o uso para aqueles que tem as versões Express ou Standard.

Uma terceira maneira, disponível a todos, é usar automação Ole com o Office. Desta maneira, podemos controlar os aplicativos Office a partir dos programas desenvolvidos em qualquer linguagem que suporte este tipo de interface, como as linguagens .net ou mesmo linguagens não gerenciadas como VB6 ou Delphi. Esta é a maneira que iremos usar para converter nossos arquivos para o novo formato. O programa irá converter arquivos do Word, porém isso pode ser facilmente modificado para converter arquivos do Excel ou PowerPoint.

Para isso, é necessário que você tenha o Office instalado na máquina de desenvolvimento. Se a versão 2007 estiver instalada, não é necessário baixar mais nada, mas se você tiver uma versão anterior do Office (a partir da versão 2000), você deve baixar e instalar o Office Compatibility Pack, para dar acesso ao novo formato de arquivos.

# Automação Ole com Word

Os aplicativos do Office expõem diversos objetos, que podem ser usados por qualquer aplicação para controlá-los, da mesma maneira que se estivessemos fazendo macros para execução dentro dos próprios aplicativos. Com nosso programa, podemos abrir o Word, abrir um documento, modificá-lo e salvá-lo, sem que o usuário tenha necessidade de tomar nenhuma ação.

Inicialmente, iremos criar um pequeno programa que abre um documento do Word. No Visual Studio, crie um novo projeto do tipo Windows Application e chame-o de **AbreWord**. Para possibilitar a automação Ole, devemos adicionar a referência à biblioteca do Word. No Solution Explorer, selecione *Add Reference* e, na guia **COM**, adicioe a referência a **Microsoft Word XX Object Library**, onde **XX** indica a versão do Office que está usando (12 é a versão 2007).

Em seguida, adicione um **OpenFileDialog**, configurando sua propriedade **Filter** para **Arquivos Word|\*.doc|Todos arquivos|\*.\***. Deixe a propriedade **FileName** em branco. Inclua um botão, mudando sua propriedade **Text** para **Abre** e adicione um manipulador para o seu evento **Click**.

Devemos adicionar o namespace **Microsoft.Office.Interop.Word** à lista de namespaces usados em nosso programa. Em seguida, devemos criar o manipulador do evento **Click** do botão:

private void button1\_Click(object sender, EventArgs e)

{

object readOnly = false;

object visible = true;

object missing = System.Reflection.Missing.Value;

object NomeArq;

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

NomeArq = openFileDialog1.FileName;

// Cria uma instância do Word

ApplicationClass wordApp = new ApplicationClass();

// Mostra o Word

wordApp.Visible = true;

// Abre o documento

wordApp.Documents.Open(ref NomeArq, ref missing, ref readOnly,

ref missing, ref missing, ref missing, ref missing, ref missing,

ref missing, ref missing, ref missing, ref visible, ref missing,

ref missing, ref missing, ref missing);

}

}

Inicialmente, mostramos a caixa de abrir arquivos. Se o usuário escolher um arquivo, criamos uma nova instância do Word, criando um objeto do tipo **ApplicationClass**. Em seguida, mostramos o Word e abrimos o documento. Como você pode ver, o valor **System.Reflection.Missing.Value** é passado para a maior parte dos parâmetros. Estes parâmetros são opcionais quando estamos usando uma macro VBA ou criando um programa em VB.net. Porém, como o C# não tem parâmetros opcionais, usamos este valor para que seja utilizado o valor padrão.

Após a execução do programa, o Word fica aberto com o documento. Se clicarmos o botão mais de uma vez, mais instâncias do Word serão abertas, cada uma com um documento diferente. O passo seguinte é salvar o documento no formato novo.

Para isso, devemos usar o método SaveAs do documento. Devemos modificar o manipulador do evento **Click** para o seguinte:

// Abre o documento

Document doc = wordApp.Documents.Open(ref NomeArq, ref missing, ref readOnly,

ref missing, ref missing, ref missing, ref missing, ref missing,

ref missing, ref missing, ref missing, ref visible, ref missing,

ref missing, ref missing, ref missing);

NomeArq = NomeArq + "x";

object Formato = WdSaveFormat.wdFormatXMLDocument;

doc.SaveAs(ref NomeArq, ref Formato, ref missing, ref missing, ref missing,

ref missing, ref missing, ref missing, ref missing, ref missing,

ref missing, ref missing, ref missing, ref missing, ref missing,

ref missing);

Guardamos o resultado do método **Open** em uma variável e adicionamos um “x” ao final do nome do arquivo. Finalmente, salvamos o arquivo, usando o formato **wdFormatXMLDocument**. Esta constante não está disponível quando usamos as versões do Word mais antigas que o Word 2007. Para isso, precisamos varrer o objeto **FileConverters** para encontrar o valor da constante correto. Se você estiver usando uma versão mais antiga, você deve colocar este código, ao invés da atribuição da constante:

object Formato = -1;

foreach (FileConverter conv in wordApp.FileConverters)

if (conv.ClassName == "MSWord12")

{

Formato = conv.SaveFormat;

break;

}

Com isso, ao executar o programa, abrimos o arquivo e o convertemos para o novo formato (o arquivo antigo não é removido e continua no disco). Para fechar o Word após a conversão usamos o método **Quit** ao final:

wordApp.Quit(ref missing, ref missing, ref missing);

Se você executar o programa e em seguida abrir o arquivo convertido, verá que ele está marcado com o modo de compatibilidade. Este modo desabilita algumas inovações do Word 2007, que não existem nas versões anteriores. Se você quiser salvar sem o modo de compatibilidade, use doc.Convert(); antes de salvar o arquivo.

Isto é melhor que usar o Word, mas ainda não é o ideal: temos que clicar no botão e selecionar o arquivo. O ideal é converter um lote de arquivos de uma vez. Faremos isso logo a seguir.

# Pesquisando arquivos

Podemos pesquisar os arquivos em um diretório usando-se a classe **DirectoryInfo**. Ela permite manipular ou enumerar os arquivos de um diretório. Esta classe está em **System.IO**. O método **GetFiles** retorna um array de elementos do tipo **FileInfo**, que traz informações sobre os arquivos encontrados.Este método retorna apenas os arquivos do diretório selecionado, e não de seus subdiretórios. Para pesquisar os subdiretórios usamos o método **GetDirectories**, que retorna os subdiretórios do diretório selecionado. Com esta lista, chamamos a função de pesquisa novamente. A função irá preencher uma List<string>. Crie um novo projeto do tipo Windows Application e chame-o de **ConverteDocs**. Adicione a referência à biblioteca do Word, da mesma maneira que fizemos no programa anterior. Na classe criada, coloque o seguinte:

private void PesquisaArquivos(string NomeDir)

{

// cria instância de DirectoryInfo para o diretório selecionado

DirectoryInfo DirInfo = new DirectoryInfo(NomeDir);

try

{

// obtém arquivos do diretório

FileInfo[] AFileInfo = DirInfo.GetFiles("\*.doc");

// processa arquivos, adicionando-os na ListView

foreach (FileInfo FilInfo in AFileInfo)

ListaArqs.Add(FilInfo.FullName);

// procura subdiretórios

DirectoryInfo[] ADirInfo = DirInfo.GetDirectories();

// chama função recursivamente

foreach (DirectoryInfo DirecInfo in ADirInfo)

PesquisaArquivos(DirecInfo.FullName);

}

catch

{

}

}

Esta é a nossa função de pesquisa, ela procura todos os arquivos do tipo **doc** no diretório selecionado e em seus subdiretórios, adicionando os nomes numa **List<string>** chamada **ListaArqs**.

Na Form, coloque uma **TextBox** na parte superior e uma **CheckedListBox** na parte inferior. Insira um componente FolderBrowserDialog e altere sua propriedade **ShowNewFolderButton** para **false**. Ao lado da TextBox, coloque um pequeno botão, alterando sua propriedade **Text** para “...” e coloque o seguinte código no manipulador do evento **Click**:

private void button1\_Click(object sender, EventArgs e)

{

folderBrowserDialog1.SelectedPath = textBox1.Text;

if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)

textBox1.Text = folderBrowserDialog1.SelectedPath;

}

Este botão mostra a caixa de díalogo para seleção de diretório e preenche a caixa de edição caso o usuário tenha selecionado algo. Em seguida, coloque mais um botão ao lado do botão de selção de diretório, alterando a propriedade **Text** para **Pesquisa**. Você deve ter algo semelhante à Figura 1.

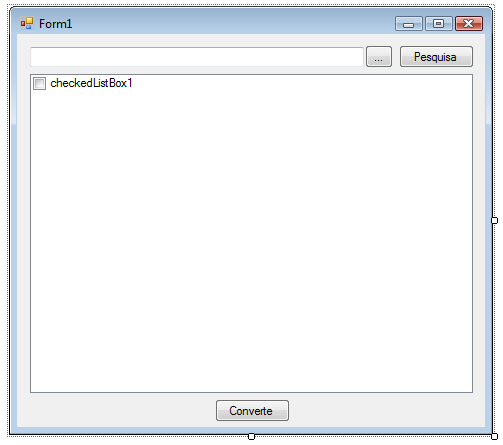


Figura 1 – Tela da pesquisa nos arquivos

O código para o manipulador do evento **Click** do botão que acabamos de inserir é:

private void button2\_Click(object sender, EventArgs e)

{

if (textBox1.Text != "")

{

ListaArqs = new List<string>();

PesquisaArquivos(textBox1.Text);

checkedListBox1.Items.Clear();

foreach(string Arq in ListaArqs)

checkedListBox1.Items.Add(Arq);

}

}

Ele verifica se a caixa de edição está preenchida e chama a função **PesquisaArquivos**. A função preenche a lista de arquivos encontrados e, em seguida, preenche a **CheckedListBox.** Esta tem checkboxes, para que o usuário possa selecionar os arquivos que deseja converter. A função de conversão é semelhante ao código do primeiro programa:

private ApplicationClass wordApp;

private void ConverteArquivo(string NomeArq)

{

object read\_only = false;

object visible = false;

object missing = System.Reflection.Missing.Value;

object Formato = WdSaveFormat.wdFormatXMLDocument;

object Arquivo = NomeArq;

Document doc = wordApp.Documents.Open(ref Arquivo, ref missing,

ref read\_only, ref missing, ref missing,

ref missing, ref missing, ref missing, ref missing,

ref missing, ref missing, ref visible, ref missing,

ref missing, ref missing, ref missing);

Arquivo = Arquivo + "x";

doc.SaveAs(ref Arquivo, ref Formato, ref missing,

ref missing, ref missing, ref missing, ref missing, ref missing,

ref missing, ref missing, ref missing, ref missing, ref missing,

ref missing, ref missing, ref missing);

}

Finalmente, coloque um botão abaixo da **CheckListBox** e mude sua propriedade **Text** para **Converte**. O código do manipulador do evento **Click**, que irá converter os arquivos, é o seguinte:

private void button3\_Click(object sender, EventArgs e)

{

object missing = System.Reflection.Missing.Value;

wordApp = new ApplicationClass();

Cursor = Cursors.WaitCursor;

try

{

foreach (string Nome in checkedListBox1.CheckedItems)

ConverteArquivo(Nome);

}

finally

{

Cursor = Cursors.Default;

wordApp.Quit(ref missing, ref missing, ref missing);

}

}

O código cria uma instância do Word, muda o cursor para uma ampulheta e converte os arquivos que têm a checkbox marcada. Finalmente, ele volta o cursor ao normal e fecha o Word.

# Conclusões

Como pudemos ver, a conversão de arquivos doc em docx é relativamente simples, usando automação Ole: abrimos o arquivo e o salvamos com um novo nome. Adicionando isto à pesquisa dos arquivos doc em um diretório, temos uma poderosa ferramenta que varre os arquivos do disco, mostrando os arquivos doc, que podem ser selecionados para conversão. Esta conversão diminui o tamanho total dos arquivos e faz que eles passem a ter uma arquitetura aberta, podendo ser abertospor programas criados para isso.