Word 文档信息获取方式

在Open Xml SDK中，WordprocessingDocument类代表了Word 文档中的一个包。要处理一份Word文档，首先需要通过文档创建一个WordprocessingDocument类的实例，然后操作这个实例从而实现对Word文档的操作。获取该实例之后，我们就能够访问包含了文本的主文档部分。每一个打开的XML包都包含若干个部分，一个WordprocessingDocument至少包含了一个主文档部分，该部分作为一个容器用来容纳Word文档中的主要文本内容，该WordprocessingDocument所代表的包也可以包含其他的一些部分。在一个Word文档中，主文档部分中的文本内容在XML包中使用WordprocessingML来标记。

要从一份Word文档创建WordprocessingDocument类的实例，我们可以调用Open（Stream，Boolean）方法，Open XML SDK中提供了好几个Open方法，我们采用具有2个参数的Open方法来获取实例，第一个参数代表一个Stream，该Stream关联了我们想要打开的Word文档，第二个参数可以取值为true或者false，代表我们打开的Stream是否是可编辑的。在我们的应用当中，由于对Word文档都是只读的，所以该参数均取值为false，表示不能修改打开的Word文档。创建WordprocessingDocument类实例的示例代码如下：

WordprocessingDocument document = WordprocessingDocument.Open(stream, false);

WordProcessingM文档的结构：

一份WordProcessingML文档的基本结构由document和body元素组成，之后再跟上一个或几个块级元素，例如p元素，代表一个段落。一个段落包含一个或几个r元素。r元素代表run，它代表一段具有共同属性的文本区域。一个run中包含一个或几个t元素，t元素包含一段文本。例如，对于只包含文本“测试文本”的Word文档来说，它的WordprocessingML 标记如下所示：

<w:document xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main">

<w:body>

<w:p>

<w:r>

<w:t>测试文本</w:t>

</w:r>

</w:p>

</w:body>

</w:document>

WordprocessingML中的各个元素在Open XML SDK里面都有相应的类与之对应，这些类都在DocumentFormat.OpenXml.Wordprocessing命名空间当中，下表列出了与document,body,p,r和t元素相对应的类：

|  |  |  |
| --- | --- | --- |
| WordprocessingML 元素 | Open XML SDK类 | 描述 |
| document | Document | 主文档部分的根元素 |
| body | Body | 段落、表格等块级元素的容器 |
| p | Paragraph | 段落 |
| r | Run | 一段具有共同属性的文本区域 |
| t | Text | 一段文本 |

要访问主文档部分的body，我们可以把存在的文档体赋值给Body引用，像如下代码所示：

WordprocessingDocument document = WordprocessingDocument.Open(stream, false);

Body body = document.MainDocumentPart.Document.Body;

操作段落

WordprocessingML中的段落

WordprocessingML中最基本的块级内容单元就是段落，段落用<p>元素进行存储。一个段落定义了一个由新行开始的独立内容区域。段落可以包含三部分信息：可选的段落属性、内联内容（比如run）,以及一些可选的校正ID，用来比较2篇文档的内容。

一个段落的属性是通过<pPr>元素指定的，段落属性包含对齐方式、边框、缩进、行距、底纹、文本方向等。下表列出了Open XML SDK中与段落操作相关的常用类与WordprocessingML元素的对应关系：

|  |  |
| --- | --- |
| WordprocessingML元素 | Open XML SDK类 |
| p | Paragraph |
| pPr | ParagraphProperties |
| r | Run |
| t | Text |

Paragraph类

Open XML SDK中的Paragraph类代表为WordprocessingML文档定义的Open XML文件格式schema中的段落（<p>）元素，使用Paragraph对象来操作WordprocessingML文档中的<p>元素。

ParagraphProperties类

在WordprocessingML中，一个段落的属性是通过段落属性（<pPr>）元素指定的，Open XML SDK中的ParagraphProperties类代表<pPr>元素。

Run类

一个Word文档中的段落一般都会包含文本，在WordprocessingML文档的Open XML 文件格式schema中，run（<r>）元素被用来界定一个文本区域。在Open XML SDK中，Run类代表<r>元素。

Text类

在<r>元素内部，是文本(<t>)元素，文本元素是组成文档内容文本的容器。在Open XML SDK中，Text类代表<t>元素。

以下代码实例化了一个Open XML SDK中的Paragraph类对象并使用该对象添加一段文本到WordprocessingML文档中：

public static void WriteToWordDoc(string filepath, string txt)

{

// 使用指定的文件路径来打开一个文档并对其进行编辑

using (WordprocessingDocument wordprocessingDocument =

WordprocessingDocument.Open(filepath, true))

{

// 实例化body对象

Body body = wordprocessingDocument.MainDocumentPart.Document.Body;

// 在一个段落中添加一段文本

Paragraph para = body.AppendChild(new Paragraph());

Run run = para.AppendChild(new Run());

run.AppendChild(new Text(txt));

}

}

上述代码执行完之后，写入WordprocessingML文档中的XML如下所示：

<w:p>

<w:r>

<w:t>测试文本</w:t>

</w:r>

</w:p>

操作表格

在WordprocessingML中另一个类型的块级内容就是表格，一个表格就是一系列排列在行和列中的段落（另一种块级内容）集合。

WordprocessingML中的表格是通过<tbl>元素定义的，该元素与HTML中的<table>标签很相似。表格元素指定了一个表格在文档中呈现的位置。

一个<tbl>元素有两个元素定义了它的属性：<tblPr>和<tblGrid>。其中<tblPr>定义了表格范围（例如样式和宽度）内的属性集合，而<tblGrid>定义了表格的网格布局。一个<tbl>元素可以包含任意非0的行数，每一行通过一个<tr>元素进行指定，每一个<tr>元素可以包含任意非0数目的单元格，每个单元格用<tc>元素进行指定。

下面的表格列出了处理表格时最常用的Open XML SDK中的类

|  |  |
| --- | --- |
| XML元素 | Open XML SDK类 |
| gridCol | GridColumn |
| tblGrid | TableGrid |
| tblPr | TableProperties |
| tc | TableCell |
| tr | TableRow |

Table类

Open XML SDK中的Table类代表为WordprocessingML文档定义的Open XML文件格式schema中的表格（<tbl>）元素，使用Table对象来操作WordprocessingML文档中的< tbl >元素。

TableProperties类

在WordprocessingML中，一个表格的属性是通过表格属性（<tblPr>）元素指定的，Open XML SDK中的TableProperties类代表<tblPr>元素。<tblPr>元素定义了表格级别的属性，使用TableProperties对象可以用来设置WordprocessingML文档中表格的属性。

TableGrid类

Open XML SDK中的TableGrid类代表为WordprocessingML文档定义的Open XML文件格式schema中的表格网格（<tblGrid>）元素，与网格列（<gridCol>）子元素一起，<tblGrid>元素定义了表格的列，并且指定了表格列中单元格的默认宽度，可以使用TableGrid对象来操作WordprocessingML文档中表格中的列。

GridColumn类

Open XML SDK中的GridColumn类代表为WordprocessingML文档定义的Open XML文件格式schema中的网格列（<gridCol>）元素。<gridCol>元素是<tblGrid>元素的子元素，它定义了WordprocessingML文档中一个表格的单个列。可以使用GridColumn类来操作WordprocessingML文档中的单独的列。

TableRow类

Open XML SDK中的TableRow类代表为WordprocessingML文档定义的Open XML文件格式schema中的表格行（<tr>）元素。<tr>元素定义了WordprocessingML文档中表格的一行，与HTML中的<tr>标签相似。一个表格的行也能够使用表格行属性（<trPr>）元素将一些样式应用到它上面。Open XML SDK中的TableRowProperties类代表<trPr>元素。

TableCell类

Open XML SDK中的TableCell类代表为WordprocessingML文档定义的Open XML文件格式schema中的表格单元格（<tcc>）元素。<tc>元素定义了WordprocessingML文档中表格单元格，与HTML中的<td>标签相似。一个表格的单元格也能够使用表格单元格属性（<tcPr>）元素将一些样式应用到它上面。Open XML SDK中的TableCellProperties类代表<trPr>元素。下面的代码展示了如何向一个文档中插入一个1行3列的表格：

public static void InsertTableInDoc(string filepath)

{

// 打开一个Word文档，并使该文档能够编辑

using (WordprocessingDocument wordprocessingDocument =

WordprocessingDocument.Open(filepath, true))

{

// 获取文档体

Body body = wordprocessingDocument.MainDocumentPart.Document.Body;

// 创建一个表格

Table tbl = new Table();

// 设置表格的样式和宽度

TableProperties tableProp = new TableProperties();

TableStyle tableStyle = new TableStyle() { Val = "TableGrid" };

// 使表格宽度占页面宽度的100%

TableWidth tableWidth = new TableWidth() { Width = "5000", Type = TableWidthUnitValues.Pct };

// 将设置的样式应用到表格

tableProp.Append(tableStyle, tableWidth);

tbl.AppendChild(tableProp);

// 添加3列到表格

TableGrid tg = new TableGrid(new GridColumn(), new GridColumn(), new GridColumn());

tbl.AppendChild(tg);

// 创建表格的一行

TableRow tr1 = new TableRow();

// 添加单元格到行中的每一列

TableCell tc1 = new TableCell(new Paragraph(new Run(new Text("1"))));

TableCell tc2 = new TableCell(new Paragraph(new Run(new Text("2"))));

TableCell tc3 = new TableCell(new Paragraph(new Run(new Text("3"))));

tr1.Append(tc1, tc2, tc3);

// 添加行到表格上.

tbl.AppendChild(tr1);

// 添加表格到文档

body.AppendChild(tbl);

}

}

以上代码执行之后会将如下的XMl写入WordprocessingML文档中：

<w:tbl>

<w:tblPr>

<w:tblStyle w:val="TableGrid" />

<w:tblW w:w="5000"

w:type="pct" />

</w:tblPr>

<w:tblGrid>

<w:gridCol />

<w:gridCol />

<w:gridCol />

</w:tblGrid>

<w:tr>

<w:tc>

<w:p>

<w:r>

<w:t>1</w:t>

</w:r>

</w:p>

</w:tc>

<w:tc>

<w:p>

<w:r>

<w:t>2</w:t>

</w:r>

</w:p>

</w:tc>

<w:tc>

<w:p>

<w:r>

<w:t>3</w:t>

</w:r>

</w:p>

</w:tc>

</w:tr>

</w:tbl>

Open XML 是可由不同平台上的多个应用程序自由实现的字处理文档、演示文稿和电子表格的开放式标准。Open XML 旨在如实表示用 Microsoft Office 应用程序定义的二进制格式进行编码的现有字处理文档、演示文稿和电子表格。使用 Open XML 的原因很简单：现在存在数以亿计的文档，但遗憾的是，这些文档中的信息与创建文档的程序紧密耦合。Open XML 标准的目的是分离由 Microsoft Office 应用程序创建的文档，以便其他应用程序可以独立于专有格式操作这些文档且不会丢失数据。

## [Open XML 包的结构](javascript:void(0))

Open XML 文件存储在 ZIP 存档中以方便打包和压缩。可以使用 ZIP 查看器来查看任何 Open XML 文件的结构。一个 Open XML 文档由多个文档部件构成。这些部件之间的关系自己存储在文档部件中。ZIP 格式支持随机访问每个部件。例如，应用程序可以将一张幻灯片从一个演示文稿中移到另一个演示文稿中，而无需分析幻灯片内容。同样地，应用程序可以删除字处 理文档中的所有注释，而不用分析文档的任何内容。

以 XML 标记的形式创建 Open XML 包中的文档部件。由于 XML 的结构为纯文本，因此您可以使用文本读取器查看文档部件的内容，也可以使用类似 XPath 的进程分析内容。

从结构上来讲，Open XML 文档是开放打包约定 (OPC) 包。如前面所述，一个包由一组文档部件组成。每个部件的部件名称由一个段序列或路径名称组成，如"/word/theme/theme1.xml"。包中 包含一个 [Content\_Types].xml 部件，通过该部件可以确定包中所有文档部件的内容类型。以 .rels 扩展名结尾的关系部件中包含一组源包或部件的显式关系。

字处理文档通过使用 WordprocessingML 标记进行描述。有关详细信息，请参阅[使用 WordprocessingML 文档 (Open XML SDK)](https://msdn.microsoft.com/zh-cn/library/office/gg278327.aspx)。一个 WordprocessingML 文档由一组文章组成，其中每篇文章属于下列项之一：

* 主文档（唯一必需的文章）
* 词汇表文档
* 页眉和页脚
* 注释
* 文本框
* 脚注和尾注

演示文稿通过使用 PresentationML 标记进行描述。有关详细信息，请参阅[使用 PresentationML 文档 (Open XML SDK)](https://msdn.microsoft.com/zh-cn/library/office/gg278318.aspx)。演示文稿包可以包含下列文档部件：

* 幻灯片母版
* 备注母版
* 讲义母版
* 幻灯片版式
* 说明

电子表格工作簿通过使用 SpreadsheetML 标记进行描述。有关详细信息，请参阅[使用 SpreadsheetML 文档 (Open XML SDK)](https://msdn.microsoft.com/zh-cn/library/office/gg278328.aspx)。工作簿包可以包含：

* 工作簿部件（必需部件）
* 一张或多张工作表
* 图表
* 表
* 自定义 XML

## [Open XML SDK 1.0](javascript:void(0))

Open XML SDK 版本 1 简化了对 Open XML 包的操作。Open XML SDK 应用程序编程接口 (API) 封装您通常对 Open XML 包所执行的许多常见任务，以便您只使用少数几行代码就可以执行复杂的操作。一些常见任务有：

* 搜索—通过几行代码，您可以在一组 Excel 2007 工作表中搜索某些任意数据。
* 文档合并—您可以通过编程方式合并现有文档的文档部件来创建文档。例如，您可以从各种 PowerPoint 2007 演示文稿中提取幻灯片来创建单个演示文稿。
* 验证—通过几行代码，您可以验证包中的文档部件，也可以对照架构验证整个包。
* 数据更新—使用 Open XML 对象模型，您可以轻松修改多个包中的数据。
* 隐私—通过几行代码，您可以在文档发布之前，从中移除注释和其他个人信息。

## [Open XML SDK 2.0 for Microsoft Office](javascript:void(0))

Open XML SDK 2.0 for Microsoft Office 将强类型类支持从版本 1.0 中提供的部件类扩展到每个部件中的 XML 内容。仍支持版本 1.0 中提供的所有函数。凭借版本 2.0，您能够针对部件内部的 XML 内容进行编程。SDK 支持采用 LINQ to XML 样式进行编程，与传统的 W3C XML DOM 编程模型相比，此编程方式使针对 XML 内容进行编码更加容易。

SDK 支持以下常见任务/方案：

* 强类型类和对象—不再需要依赖一般 XML 功能操作 XML（这需要您了解元素/属性/值拼写以及命名空间），通过使用 Open XML SDK，您只需操作表示元素/属性/值的对象即可实现相同的解决方案。所有架构类型都表示为强类型的公共语言运行时 (CLR) 类，所有属性值都表示为枚举。
* 内容构造、搜索和操作—LINQ 技术直接内置到 SDK 中。因此，您可以直接对表示 Open XML 元素的对象执行功能构造和 lambda 表达式查询。此外，SDK 还提供对对象集合（如表和段落）的支持，从而允许您轻松地遍历和操作内容。
* 验证—Open XML SDK 2.0 for Microsoft Office 提供验证功能，从而使您能够对照 Open XML 格式的不同验证方式来验证 Open XML 文档。

## [Open XML SDK 2.5 for Office](javascript:void(0))

Open XML SDK 2.5 提供支持 Microsoft Office 2013 的命名空间和成员。Open XML SDK 2.5 还可读取 ISO/IEC 29500 严格格式文件。严格格式是不包括旧功能的过渡格式的子集 - 在理论上，这使得新的实施者更容易提供支持，因为该格式具有较小的技术涉及面。

SDK 支持以下常见任务/方案：

* 支持 Office 2013 Preview 文件格式—除了 Microsoft Office 类的 Open XML SDK 2.0，Open XML SDK 2.5 还提供了新的类，利用这些类，您可以编写和生成用于处理新的 Office 2013 功能的 Open XML 文件扩展名的应用程序。
* 读取 ISO 严格文档文件—Open XML SDK 2.5 可读取 ISO/IEC 29500 严格格式文件。当 Open XML SDK 2.5 API 打开严格格式文件时，通过将 http://purl.oclc.org/ooxml/ 命名空间映射到对应的 http://schemas.openxmlformats.org/ 命名空间，文件中的每个 Open XML 部件被加载到 Open XML SDK 2.5 的 OpenXmlPart 类。
* 修复 Open XML SDK 2.0 for Microsoft Office—Open XML SDK 2.5 包括在 Open XML SDK 2.0 for Microsoft Office 中对已知问题的修复。这些已知问题包括 PowerPoint 演示文稿中丢失的空白和 Word 文档（其中指定参数被报告为超出有效值范围）中有关自定义 UI 的问题。