**Word动态模块填充之DataRegion篇**

**1、功能介绍**

PageOffice能实现在在线打开的Word文档中指定的位置动态填充数据、修改Word中文本的样式、提交Word文档中某些数据等功能。

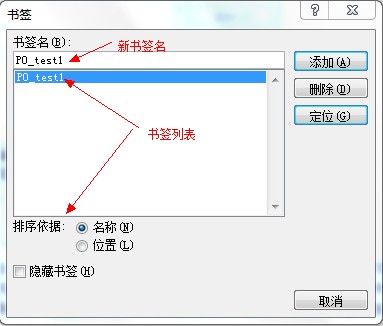
在线编辑Word时，可能需要在Word文档中某些位置动态填充某些数据，例如，在某个套红文件中想要动态填充文档的标题、发文号、发文人、发文时间等等；在线编辑Word文档时，还有可能会想要在满足某些条件时改变Word文档中某部分数据的值或者是调整某部分Word文档的颜色样式；另外，在线编辑Word文档时，还可能会像提交HTML页面中的表单一样，想要将当前Word文档中的某些数据提交到其他页面中。

而通过PageOffice开发平台下的数据区域（即DataRegion类），上述的功能全部都能轻松实现，而且实现方法十分简单和易懂。

什么是数据区域？数据区域是Word文档中具有 "PO\_" 前缀命名的书签所定位的文档区域。简单来说，数据区域就是一种特殊的Word书签对象，便于定位Word文档中的内容。数据区域是PageOffice开发平台中重要的技术概念。

一般的开发人员使用Word比较少，可能不清楚Word书签是怎样插入的，下面简单的介绍一下插入书签的方法。

第一种方法：把光标定位到需要标记数据位置的地方，点Word菜单中的“插入”-“书签”，就会弹出一个标题为“书签”的对话框（如下图所示），输入新书签的名称，注意：书签名必须以字母、汉字、中文标点等开头，可以包含数字但中间不能有空格（用PageOffice开发的时候不推荐使用中文命名书签名）。点右侧的“添加”按钮，新的书签名将出现在下面的列表中。



第二种方法：选择几个文字、或一段文字、或者选择一段包含表格和图片的内容，为选中的内容指定书签的对象，然后执行第一种方法的同样操作，“插入”→“书签”……

注意：如果新插入位置或新对象采用的是已有的书签名，原有的书签将自动取消。

PageOffice示例代码中的模板在制作的时候，多采用第二中方法定义书签，插入书签之前会先写一个标示性的词语用中括号括起来，比如：[姓名]，然后选中“[姓名]”，再插入书签。这样做的目的是便于在查看或编辑模板数据位置的时候一目了然。

在使用PageOffice开发的过程中，为了避免出现与用户自己定义的书签出现冲突，要求插入的书签名称必须以“PO\_”开头。注意是字母o，不是数字0。

在PageOffice的概念里提到的数据区域，本质上就是书签，但是只有“PO\_”（或者小写的“po\_”）开头的书签才叫数据区域，请大家注意这点。

**2、数据区域的具体使用方法**

数据区域的填充、数据区域格式的控制、及数据区域的提交等都是通过DataRegion类实现的，很多Word操作都需要调用DataRegion类完成。WordDocument类对象能通过调用 OpenDataRegion(String) 方法或 CreateDataRegion(String, DataRegionInsertType, String) 方法获得 DataRegion 类对象：

DataRegion dataRegion1 = wordDoc.OpenDataRegion("PO\_userName"); // OpenDataRegion方法中的参数代表Word文档中插入的书签名称

PageOffice.WordWriter.DataRegion title = doc.CreateDataRegion("PO\_title",PageOffice.WordWriter.DataRegionInsertType.After, "[home]");//创建DataRegion对象，三个参数分别代表新插入书签的名称、新书签的插入位置、相关联的书签名称（“[home]”代表Word文档中最开始的光标位置，“[end]”代表Word文档中的结束光标位置）

DataRegion类与WordDocument类、Font类、ParagraphFormat类等结合使用可实现许多功能。

下面介绍一下DataRegion类中常用的几项功能。

①数据区域的动态填充。以在待打开的Word文档中指定位置显示某个人员的姓名和所在部门为例，具体实现方法为：

先手动打开Word文档，在Word中指定位置插入书签“PO\_userName”和“PO\_deptName”，然后保存Word文档。然后调用WordDocument类对象的OpenDataRegion(String)方法获取DataRegion类对象，通过DataRegion类对象的Value属性给数据区域赋值，再通过PageOfficeCtrll类的WebOpen(String, OpenModeType, String)方法在线打开Word文档，就可在Word中显示动态填充的人员姓名和所在部门了。具体实现代码如下：

//Java实现代码

PageOfficeCtrl poCtrl1 = **new** PageOfficeCtrl(request);

poCtrl1.setServerPage(request.getContextPath()+"/poserver.zz"); //设置服务页面

WordDocument doc = **new** WordDocument();

//打开数据区域 openDataRegion(string）方法中的参数为Word中的书签名称

DataRegion dataRegion1 = doc.openDataRegion("PO\_userName");

//给数据区域赋值

dataRegion1.setValue("张三");

DataRegion dataRegion2 = doc.openDataRegion("PO\_deptName");

dataRegion2.setValue("销售部");

... ...

poCtrl1.setWriter(doc); //不要忘记此行

poCtrl1.webOpen("doc/test.doc", OpenModeType.docNormalEdit, "张三");//打开Word文件

poCtrl1.setTagId("PageOfficeCtrl1"); //此行必须

//C#实现代码

PageOfficeCtrl1.ServerPage = Request.ApplicationPath + "/pageoffice/server.aspx";//设置服务页面

PageOffice.WordWriter.WordDocument wordDoc = new PageOffice.WordWriter.WordDocument();

//打开数据区域，OpenDataRegion方法的参数代表Word文档中的书签名称

PageOffice.WordWriter.DataRegion dataRegion1 = wordDoc.OpenDataRegion("PO\_userName");

//为DataRegion赋值

dataRegion1.Value = "张三";

PageOffice.WordWriter.DataRegion dataRegion2 = wordDoc.OpenDataRegion("PO\_deptName");

dataRegion2.Value = "销售部";

... ...

PageOfficeCtrl1.SetWriter(wordDoc); //不要忘记此行

PageOfficeCtrl1.WebOpen("doc/test.doc", PageOffice.OpenModeType.docNormalEdit, "文档操作人姓名");//打开Word文档

代码显示效果如图1所示：

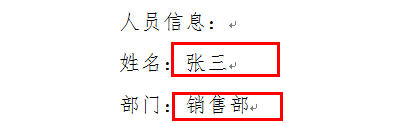


图1

上述示例的完整代码请参考相应开发语言示的Samples示例包中基础功能的第11个：“给Word文档中的数据区域（DataRegion）赋值的简单示例”。

②数据区域格式的控制。

DataRegion类结合Font类使用可控制数据区域内文本的字体样式。例如，将Word文档中人员的姓名设置为：隶书、24磅、蓝色、加粗，将部门名称设置为红色，具体的实现代码如下：

//Java实现代码

... ...

WordDocument doc = **new** WordDocument();

//打开数据区域

DataRegion dataRegion1 = doc.openDataRegion("PO\_userName");

//给数据区域赋值

dataRegion1.setValue("张三");

//设置字体样式

dataRegion1.getFont().setColor(Color.blue);

dataRegion1.getFont().setSize(24);

dataRegion1.getFont().setName("隶书");

//body.getFont().setNameAscii("Times New Roman");//设置英文字符的字体名称

dataRegion1.getFont().setBold(**true**);

//还可以先定义Font类对象，然后在进行自体的设置

//Font font =dataRegion1.getFont();

//font.setColor(Color.blue);

... ...

DataRegion dataRegion2 = doc.openDataRegion("PO\_deptName");

dataRegion2.setValue("销售部");

dataRegion2.getFont().setColor(Color.red);

poCtrl1.setWriter(doc);

... ...

poCtrl1.webOpen("doc/test.doc", OpenModeType.docNormalEdit, "张三");//打开Word文件

poCtrl1.setTagId("PageOfficeCtrl1"); //此行必须

代码的运行效果如图2所示。

//C#实现代码

PageOfficeCtrl1.ServerPage = Request.ApplicationPath + "/pageoffice/server.aspx";//设置服务器页面

PageOffice.WordWriter.WordDocument wordDoc = new PageOffice.WordWriter.WordDocument();

//打开数据区域，OpenDataRegion方法的参数代表Word文档中的书签名称

PageOffice.WordWriter.DataRegion dataRegion1 = wordDoc.OpenDataRegion("PO\_userName");

//为DataRegion赋值

dataRegion1.Value = "张三";

//设置字体样式

dataRegion1.Font.Color = Color.Blue;

dataRegion1.Font.Size = 24f;

dataRegion1.Font.Name = "隶书";

//dataRegion1.Font.NameAscii = "Times New Roman";//设置英文字符的字体名称

dataRegion1.Font.Bold = true;

//PageOffice.WordWriter.Font font = title.Font;

//font.Color = Color.Blue;

... ...

PageOffice.WordWriter.DataRegion dataRegion2 = wordDoc.OpenDataRegion("PO\_deptName");

dataRegion2.Value = "销售部";

dataRegion2.Font.Color = Color.Red;

PageOfficeCtrl1.SetWriter(wordDoc); //不要忘记此行

PageOfficeCtrl1.WebOpen("doc/test.doc", PageOffice.OpenModeType.docNormalEdit, "文档操作人姓名");//打开Word文档

代码的运行效果如图2所示：

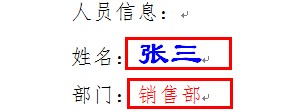


图2

上述示例的完整代码请参考相应开发语言的Samples示例包中高级功能中的第27个：“给Word文档中的数据区域（DataRegion）赋值并设置样式”。

另外，DataRegion类结合ParagraphFormat类使用还可控制文本段落的样式。例如设置标题所在段落设置为多倍行间距、居中对齐；正文内容第一段设置为最小行间距、左对齐、首行缩进21磅；第二段设置为1.5倍行间距、左对齐、首行缩进21磅；第三段设置为居中对齐。具体的实现代码如下：

//Java实现代码

WordDocument doc = **new** WordDocument();

DataRegion title = doc.createDataRegion("PO\_title",

DataRegionInsertType.After, "[home]"); //创建数据区域

title.setValue("C#中Socket多线程编程实例\n");//给DataRegion对象赋值

//设置字体：粗细、大小、字体名称、是否是斜体

... ...

//定义段落对象

ParagraphFormat titlePara = title.getParagraphFormat();

//设置段落对齐方式

titlePara.setAlignment(WdParagraphAlignment.wdAlignParagraphCenter);

//设置段落行间距

titlePara.setLineSpacingRule(WdLineSpacing.wdLineSpaceMultiple);

//设置内容

//第一段

DataRegion body = doc.createDataRegion("PO\_body",DataRegionInsertType.After, "PO\_title");

//设置字体：粗细、是否是斜体、大小、字体名称、字体颜色

... ...

//给DataRegion对象赋值

body.setValue("是微软随着VS.net新推出的一门语言。它作为一门新兴的语言，有着C++的强健，又有着VB等的RAD特性。而且，微软推出C#主要的目的是为了对抗Sun公司的Java。大家都知道Java语言的强大功能，尤其在网络编程方面。于是，C#在网络编程方面也自然不甘落后于人。本文就向大家介绍一下C#下实现套接字（Sockets）编程的一些基本知识，以期能使大家对此有个大致了解。首先，我向大家介绍一下套接字的概念。\n");

ParagraphFormat bodyPara = body.getParagraphFormat();

bodyPara.setLineSpacingRule(WdLineSpacing.wdLineSpaceAtLeast);

bodyPara.setAlignment(WdParagraphAlignment.wdAlignParagraphLeft);

//设置段落首行缩进,以磅为单位

bodyPara.setFirstLineIndent(21);

//第二段

... ...

代码的运行效果如图3所示。

//C#实现代码

PageOffice.WordWriter.DataRegion title = doc.CreateDataRegion("PO\_title", PageOffice.WordWriter.DataRegionInsertType.After, "[home]"); //创建数据区域

title.Value = "C#中Socket多线程编程实例\n";

//设置字体：粗细、大小、字体名称、是否是斜体

... ...

//定义段落对象

PageOffice.WordWriter.ParagraphFormat titlePara = title.ParagraphFormat;

//设置段落对齐方式

titlePara.Alignment = PageOffice.WordWriter.WdParagraphAlignment.wdAlignParagraphCenter;

//设置段落行间距

titlePara.LineSpacingRule = PageOffice.WordWriter.WdLineSpacing.wdLineSpaceMultiple;

//设置内容

//第一段

PageOffice.WordWriter.DataRegion body = doc.CreateDataRegion("PO\_body", PageOffice.WordWriter.DataRegionInsertType.After, "PO\_title");

//设置字体：粗细、是否是斜体、大小、字体名称、字体颜色

... ...

//给DataRegion对象赋值

body.Value = "是微软随着VS.net新推出的一门语言。它作为一门新兴的语言，有着C++的强健，又有着VB等的RAD特性。而且，微软推出C#主要的目的是为了对抗Sun公司的Java。大家都知道Java语言的强大功能，尤其在网络编程方面。于是，C#在网络编程方面也自然不甘落后于人。本文就向大家介绍一下C#下实现套接字（Sockets）编程的一些基本知识，以期能使大家对此有个大致了解。首先，我向大家介绍一下套接字的概念。\n";

//创建ParagraphFormat对象

PageOffice.WordWriter.ParagraphFormat bodyPara = body.ParagraphFormat;

bodyPara.LineSpacingRule = PageOffice.WordWriter.WdLineSpacing.wdLineSpaceAtLeast;

bodyPara.Alignment = PageOffice.WordWriter.WdParagraphAlignment.wdAlignParagraphLeft;

bodyPara.FirstLineIndent = 21; //设置首行缩进，以磅为单位

//第二段

... ...

代码运行的效果如图3所示：

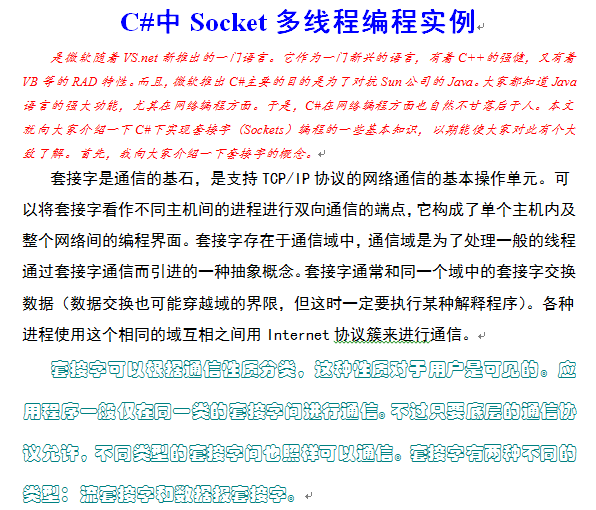


图3

上述示例的完整代码请参考Samples示例包中综合演示的第7个示例：“完全编程实现动态生成Word文件（专业版、企业版）”。

③数据区域的提交。数据区域的值是可以以类似表单的形式提交的。普通的表单提交到某个页面后，可以通过表单的名称在该页面中获取表单中<input>等控件的值，而数据区域也有着类似的功能，也可以将数据区域的值提交到某个页面，在该页面中亦可以通过数据区域的名称获取数据区域中提交的值。

具体的实现代码如下：

//Java实现代码

提交数据区域的页面“DataRegionFill.jsp”中的代码：

PageOfficeCtrl poCtrl1 = **new** PageOfficeCtrl(request);

poCtrl1.setServerPage(request.getContextPath()+"/poserver.zz"); //此行必须

WordDocument doc = **new** WordDocument();

//设置要提交的数据区域

DataRegion dataRegion1 = doc.openDataRegion("PO\_userName");

dataRegion1.setValue("张三");

DataRegion dataRegion2 = doc.openDataRegion("PO\_deptName");

dataRegion2.setValue("销售部");

poCtrl1.setWriter(doc); //不要忘记此行

poCtrl1.setSaveDataPage("SaveData.jsp");//设置保存页，即处理提交的数据区域的页面

//打开Word文件

poCtrl1.webOpen("doc/test.doc", OpenModeType.docSubmitForm, "张三");

poCtrl1.setTagId("PageOfficeCtrl1"); //此行必须

处理提交的数据区域的页面“SavaData.jsp”中的代码：

WordDocument doc = **new** WordDocument(request, response);

//获取提交的数据区域对象，openDataRegion 方法中的参数为提交的数据区域名称

DataRegion dataRegion1 = doc.openDataRegion("PO\_userName");

//获取提交的数据区域的值

String dataRegionValue1 = dataRegion1.getValue();

DataRegion dataRegion2 = doc.openDataRegion("PO\_deptName");

String dataRegionValue2 = dataRegion2.getValue();

doc.close();//不要忘记此行

//C#实现代码

提交数据区域的页面“DataRegionFill.aspx”中的代码：

PageOffice.WordWriter.WordDocument wordDoc = new PageOffice.WordWriter.WordDocument();

//设置要提交的数据区域

PageOffice.WordWriter.DataRegion dataRegion1 = wordDoc.OpenDataRegion("PO\_userName");

dataRegion1.Value = "张三";

wordDoc.OpenDataRegion("PO\_deptName").Value = "人事部";

PageOfficeCtrl1.SetWriter(wordDoc);// 注意不要忘记此代码，如果缺少此句代码，不会赋值成功。

//设置服务器页面

PageOfficeCtrl1.ServerPage = Request.ApplicationPath + "/pageoffice/server.aspx";

PageOfficeCtrl1.SaveDataPage = "SavaData.aspx";//设置保存页，即处理提交的数据区域的页面

PageOfficeCtrl1.WebOpen("doc/test.doc", PageOffice.OpenModeType.docNormalEdit, "文档操作人姓名");

处理提交的数据区域的页面“SavaData.aspx”中的代码：

WordDocument doc = new WordDocument();

DataRegion dataRegion1 = doc.OpenDataRegion("PO\_userName");//获取提交的数据区域对象

string dataRegionValue1 = dataRegion1.Value;//获取提交的数据区域的值

DataRegion dataRegion2 = doc.OpenDataRegion("PO\_deptName");

string dataRegionValue2 = dataRegion2.Value;

doc.Close();//不要忘记此行

**3、DataRegion与DataTag的区别**

Pageoffice开发平台下，还有另一个可以在Word中指定位置动态填充数据的类：DataTag。那么DataRegion与DataTag的区别在哪呢？

DataRegion类是要与书签结合使用的，既可以先在Word文档中插入以“PO\_”开头的书签定义数据区域，再在程序中通过WordDocument类对象的OpenDataRegion(String DataRegionName)方法获取DataRegion对象；又可以通过WordDocument类对象的CreateDataRegion(String NewDataRegionName, DataRegionInsertType insertType, String RelativeDataRegionName)方法直接创建数据区域从而获取DataRegion类对象。而DataTag类是与Word文档中用户插入的自定义特征文本结合使用的。在程序中DataTag类对象只能通过OpenDataTag(String DataTagName)方法获取。

DataTag的具体使用方法此处不再详细介绍，请参考相应开发语言的开发帮助和示例包，以及相关的介绍文章：Word动态模版填充之DataTag篇。

另外，数据区域的名称（即书签名称）必须以“PO\_”开头，两个书签之间要有最少一个的有效字符，不能重复，数据区域的名称和Word文档中的位置是一一对应的。而数据标签的名称就是具有一定特征的Word字符串文本，便于定位Word文档中的内容，可以重复，数据标签的名称和Word文档中的位置是一对多的关系。也就是说，同一个Name的DataRegion只能表示Word文档中唯一一个位置，而在Word文档中相同的特征字符串可以在多处出现。

所以DataRegion的优点是：DataRegion既可以填充文档内容，还可以创建文档内容、提交用户在DataRegion里输入的内容、获取用户提交的DataRegion里的内容，而DataTag只能用于填充文档内容。DataRegion的缺点是：DataRegion无法简单重复，必须要设置不同的书签，而DataTag可以在文档中多处生成需要重复的内容。

而应该用DataTag还是用DataRegion，则应该视具体的需求而定。当用户要提交Word中某部分数据或者想要获取Word中提交的某部分数据时则必须要用DataRegion；当用户不需提交数据或者获取数据，并且填充的数据无重复项时，则两者均可；当用户不需提交数据或者获取数据，且填充的数据有多个重复项时，虽然两者均可，但是为了编程简便、代码高效，建议使用DataTag。