# 将Html导出为Docx

虽然在几年前就完成了[题库系统](http://shiba.hpe.sh.cn/jiaoyanzu/WULI/Tiku)，并使用[MathJax](https://www.mathjax.org/)实现了数学公式在网页上的显示，但题库的另一个重要功能——试卷的导出功能一直未能实现，主要的困难在于如何将MathJax公式转换为word的公式。

曾经试过[pandoc](http://www.pandoc.org/)，先将Html转换为Markdown，再将Markdown转换为docx能够成功，但很多细节不是很完美，特别是没能解决图像和表格的问题。

也曾研究过[Aspose.Words](https://www.aspose.com/products/words)，但未付费用户有功能限制，没有深入研究下去。

高考之后有了点空闲时间，花了大约三个星期实现了题库的导出功能。

第一个星期构造前端，使用sessionStorage处理购物篮，主要的参考文章是[使用jQuery和Session storage构建客户端购物车系统](http://www.htmleaf.com/ziliaoku/qianduanjiaocheng/201503201552.html)；使用[Sortable](http://rubaxa.github.io/Sortable/)插件实现了题目的拖动功能。

后两个星期构造后端，使用的开发工具为Visual Studio Community 2017，框架为Asp.Net MVC，核心组件为[Open XML SDK 2.7](https://github.com/OfficeDev/Open-Xml-Sdk)和[Open-Xml-PowerTools 4.3](https://github.com/OfficeDev/Open-Xml-PowerTools)。

要把Html转换为docx，**Open-Xml-PowerTools**已经做得足够完美，其中的一个类[HtmlToWmlConverterCore](https://github.com/OfficeDev/Open-Xml-PowerTools/blob/vNext/OpenXmlPowerTools/HtmlToWmlConverterCore.cs)可以将绝大多数Html标签——“a、b、body、caption、div、em、h1、h2、h3、h4、h5、h6、h7、h8、hr、html、i、blockquote、img、li、ol、p、s、span、strong、style、sub、sup、table、tbody、td、th、tr、u、ul、br、tt、code、kbd、samp、pre、article、hgroup、nav、section、dd、dl、dt、figure、main、abbr、bdi、bdo、cite、data、dfn、mark、q、rp、rt、ruby、small、time、var、wbr”转换为对应的word格式，而我主要要做的就是如何处理数学公式。流程图如下：

但在这之前，你最好去看一下[Open XML SDK 2.5](https://msdn.microsoft.com/zh-cn/library/office/bb448854.aspx)官网的一系列文章，对docx文件的构造、基本操作有所了解。由于Word文件本质上是XML文件，你需要能比较熟练的操作XML，推荐使用[LINQ to XML](https://msdn.microsoft.com/zh-cn/library/bb387098.aspx)进行这种操作。Open-Xml-PowerTools的作者Eric White的[博客](http://www.ericwhite.com/)也很不错。

想要更深入地了解docx的结构和对应的OpenXML SDK指令，那么[Open XML SDK 2.5 Productivity Tool](https://www.microsoft.com/en-us/download/details.aspx?id=30425)就是必不可少的。下面是这个工具的截图：



## 第1步 将LaTex转换为MathML

题库的内容格式为Html，公式保存为LaTex格式，例如的LaTex格式在题库中保存为“\(\sqrt 2 \)\”，我没找到.Net平台上实现此功能的类库，因此只能在前台借助于MathJax进行这个工作了。

我使用的MathJax的设置为“TeX-MML-AM\_CHTML”，因此这个公式在网页中显示为：

<span class="MathJax\_Preview" style="color: inherit;"></span>

<span id="MathJax-Element-1-Frame" class="mjx-chtml MathJax\_CHTML" tabindex="0" data-mathml="<math xmlns="http://www.w3.org/1998/Math/MathML"><msqrt><mn>2</mn></msqrt>

 </math>" role="presentation" style="font-size: 101%; position: relative;">

 <span id="MJXc-Node-4" class="mjx-math" role="math" aria-hidden="true">

 <span id="MJXc-Node-5" class="mjx-mrow">

 <span id="MJXc-Node-6" class="mjx-msqrt">

 <span class="mjx-box" style="padding-top: 0.045em;">

 <span class="mjx-surd">

 <span class="mjx-char MJXc-TeX-main-R" style="padding-top: 0.518em; padding-bottom: 0.518em;">√</span>

 </span>

 <span class="mjx-box" style="padding-top: 0.13em; border-top: 1.2px solid;">

 <span id="MJXc-Node-7" class="mjx-mrow">

 <span id="MJXc-Node-8" class="mjx-mn">

 <span class="mjx-char MJXc-TeX-main-R" style="padding-top: 0.389em; padding-bottom: 0.325em;">2</span>

 </span>

 </span>

 </span>

 </span>

 </span>

 </span>

 </span><span class="MJX\_Assistive\_MathML" role="presentation">

 <math xmlns="http://www.w3.org/1998/Math/MathML">

 <msqrt>

 <mn>2</mn>

 </msqrt>

 </math>

 </span>

</span>

<script type="math/tex" id="MathJax-Element-2">\sqrt 2 </script>

竟然有这么多！仔细观察这个结构，发现要获取公式的MathML格式，只需提取**id**以“MathJax-Element-”开头的span元素中的data-mathml属性的内容即可，然后删除这一大堆span和script标签。这个任务可以使用jquery完成，具体代码略。

经过以上操作，就可以把包含**<math>……</math>**的网页内容传递到后台了。对应的MathML代码为：

<math xmlns="http://www.w3.org/1998/Math/MathML">

<msqrt>

<mn>2</mn>

</msqrt>

</math>

## 第2步 将Html转换为XML格式

传递后台用的是post形式，因此从Request.Form中就可以得到html的字符串，但在将这个html交给**Open-Xml-PowerTools**处理前需要将它转换为XML格式，这里用的是一个类库——[html-agility-pack](https://github.com/zzzprojects/html-agility-pack)，这个库还可以方便地解析和操作XML文件。但这个库不是必须的，用**XElement.Parse**方法和**LINQ To XML**可以完成同样的工作。代码如下：

string sourceHtml = Request.Form["html"];

// 使用HtmlAgilityPack将html转换为格式良好的xml

HtmlDocument hdoc = new HtmlDocument();

hdoc.LoadHtml(exers);

hdoc.OptionOutputAsXml = true;

StringBuilder sbXml=new StringBuilder();

using (StringWriter writer = new StringWriter(sbXml))

{

 hdoc.Save(writer);

}

// 对一些特殊字符进行转义

sbXml.Replace("&amp;", "&");

sbXml.Replace("&nbsp;", "\xA0");

sbXml.Replace("&quot;", "\"");

sbXml.Replace("&lt;", "~lt;");

sbXml.Replace("&gt;", "~gt;");

sbXml.Replace("&#", "~#");

sbXml.Replace("&", "&amp;");

sbXml.Replace("~lt;", "&lt;");

sbXml.Replace("~gt;", "&gt;");

sbXml.Replace("~#", "&#");

## 第3步 将MathML转换为OMML

由于Open-Xml-PowerTools不会处理<math>标签，我也不想动它的源代码（实际上是因为源代码太复杂，我只能看懂个大概），因此需要在Open-Xml-PowerTools之前进行转换工作。

思路是：将xml中的所有<math>标签提取出来，然后将它们转换为Word中数学公式的格式——OMML（Office math markup language），在原来<math>的位置随便替换为另一个标签，其中的内容最好不会与其他内容发生冲突，我用的是“**<b>////</b>**”。

那么如何将MathML转换为OMML？答案是使用一个转换文件——MML2OMML.xsl，这个文件是office自带的，位于目录：**%ProgramFiles%\Microsoft Office\Office12\**之下（若你用的是office 2016，则在%ProgramFiles%\Microsoft Office\Office16\目录）。

不知你是否知道，将MathML公式以文本的形式粘贴到Word中时，它会自动变成Word公式，这个操作的背后就是MML2OMML.xsl在起作用。同样的目录下还有一个文件OMML2MML.xsl，它的作用是反过来转换，我们这里用不到。

的OMML代码为：

<m:oMath xmlns:m="http://schemas.openxmlformats.org/officeDocument/2006/math" xmlns:mml="http://www.w3.org/1998/Math/MathML">

 <m:rad>

 <m:radPr>

 <m:degHide m:val="on" />

 </m:radPr>

 <m:deg />

 <m:e>

 <m:r>

 <m:t>2</m:t>

 </m:r>

 </m:e>

 </m:rad>

</m:oMath>

具体的实现过程参考了这个网址：<https://stackoverflow.com/questions/10993621/openxml-sdk-and-mathml>。代码如下：

// 使用HtmlAgilityPack对xml进行进一步处理

hdoc.LoadHtml(sbXml.ToString());

// 提取xml中的所有<math></math>标签

HtmlNodeCollection mathMLs = hdoc.DocumentNode.SelectNodes("//math");

// 将结果保存在ommls集合中

List<DocumentFormat.OpenXml.Math.OfficeMath> ommls = new List<DocumentFormat.OpenXml.Math.OfficeMath>();

if (mathMLs != null)

{

 // MML2OMML.xsl文件位于目录：%ProgramFiles%\Microsoft Office\Office12\

 XslCompiledTransform xslTransform = new XslCompiledTransform();

 xslTransform.Load("../../MML2OMML.xsl");

 foreach (HtmlNode math in mathMLs)

 {

 // 将MathML转换为OMML(Office math markup language)

 using (TextReader tr = new StringReader(math.OuterHtml))

 {

 using (XmlReader reader = XmlReader.Create(tr))

 {

 using (MemoryStream ms1 = new MemoryStream())

 {

 XmlWriterSettings settings = xslTransform.OutputSettings.Clone();

 settings.ConformanceLevel = ConformanceLevel.Fragment;

 settings.OmitXmlDeclaration = true;

 XmlWriter xw = XmlWriter.Create(ms1, settings);

 xslTransform.Transform(reader, xw);

 ms1.Seek(0, SeekOrigin.Begin);

 StreamReader sr = new StreamReader(ms1, Encoding.UTF8);

 ommls.Add(new DocumentFormat.OpenXml.Math.OfficeMath(sr.ReadToEnd()));

 }

 }

 }

 }

 // 将<math></math>中的内容替换为"<b>////<b>"

 HtmlNode newNode = HtmlNode.CreateNode("<b>////</b>");

 foreach (HtmlNode math in mathMLs)

 {

 math.ParentNode.ReplaceChild(newNode, math);

 }

}

## 第4步 使用Open-Xml-PowerTools将XML转换为Docx

Open-Xml-PowerTools中的**HtmlToWmlConverter**类负责转换工作。代码如下：

// HtmlToWmlConverter要求元素不能有命名空间

XElement html = (XElement)ConvertToNoNamespace(XElement.Parse(hdoc.DocumentNode.OuterHtml));

// ConvertHtmlToWml方法的第2个参数

string styleInHtml = (string)html.Descendants().FirstOrDefault(d => d.Name.LocalName.ToLower() == "style");

styleInHtml = styleInHtml.Replace("//&lt;![CDATA[", "");

styleInHtml = styleInHtml.Replace("//]]&gt;//", "");

string authorCss = HtmlToWmlConverter.CleanUpCss(styleInHtml);

// ConvertHtmlToWml方法的第3个参数

string userCss = @"h1 { background-color: #00ff00; }";

// 使用OpenXml PowerTools将html转换为word

HtmlToWmlConverterSettings c\_settings = HtmlToWmlConverter.GetDefaultSettings();

// 设置包含图片文件的路径

c\_settings.BaseUriForImages = "../../";

// 默认的字体大小为14pt，即小四号，而中文宋体五号字对应10.5pt

c\_settings.DefaultFontSize = 10.5d;

// emptyDocument也可以不设置，即在ConvertHtmlToWml方法中的将这个参数设置为null，HtmlToWmlConverter会自动在内部创建一个默认的WmlDocument实例

WmlDocument emptyDocument = new WmlDocument("../../template.docx");

WmlDocument doc = HtmlToWmlConverter.ConvertHtmlToWml(defaultCss, authorCss, userCss, html, c\_settings, emptyDocument, null);

其中调用的**ConvertToNoNamespace**方法的代码如下：

private static object ConvertToNoNamespace(XNode node)

{

 XElement element = node as XElement;

 if (element != null)

 {

 return new XElement(element.Name.LocalName,

 element.Attributes().Where(a => !a.IsNamespaceDeclaration),

 element.Nodes().Select(n => ConvertToNoNamespace(n)));

 }

 return node;

}

稍微解释一下**ConvertHtmlToWml**方法的前三个参数，它们都是包含css的字符串，其中defaultCss的代码如下：

static string defaultCss =

 @"html, address,

blockquote,

body, dd, div,

dl, dt, fieldset, form,

frame, frameset,

h1, h2, h3, h4,

h5, h6, noframes,

ol, p, ul, center,

dir, hr, menu, pre { display: block; unicode-bidi: embed }

li { display: list-item }

head { display: none }

table { display: table; }

tr { display: table-row }

thead { display: table-header-group }

tbody { display: table-row-group }

tfoot { display: table-footer-group }

col { display: table-column }

colgroup { display: table-column-group }

td, th { display: table-cell; }

caption { display: table-caption }

th { font-weight: bolder; text-align: center }

caption { text-align: center }

body { margin: auto; }

h1 { font-size: 16pt; margin: auto; }

h2 { font-size: 14pt; margin: auto; }

h3 { font-size: 1.17em; margin: auto; }

h4, p,

blockquote, ul,

fieldset, form,

ol, dl, dir,

menu { margin: auto }

a { color: blue; }

h5 { font-size: .83em; margin: auto }

h6 { font-size: .75em; margin: auto }

h1, h2, h3, h4,

h5, h6, b,

strong { font-weight: bolder }

blockquote { margin-left: 40px; margin-right: 40px }

i, cite, em,

var, address { font-style: italic }

pre, tt, code,

kbd, samp { font-family: monospace }

pre { white-space: pre }

button, textarea,

input, select { display: inline-block }

big { font-size: 1.17em }

small { font-size: .83em }

sub { vertical-align: sub }

sup { vertical-align: super }

table { border-spacing: 2px; }

thead, tbody,

tfoot { vertical-align: middle }

td, th, tr { vertical-align: inherit }

s, strike, del { text-decoration: line-through }

hr { border: 1px inset }

ol, ul, dir,

menu, dd { margin-left: 40px }

ol { list-style-type: decimal }

ol ul, ul ol,

ul ul, ol ol { margin-top: 0; margin-bottom: 0 }

u, ins { text-decoration: underline }

br:before { content: ""\A""; white-space: pre-line }

center { text-align: center }

:link, :visited { text-decoration: underline }

:focus { outline: thin dotted invert }

/\* Begin bidirectionality settings (do not change) \*/

BDO[DIR=""ltr""] { direction: ltr; unicode-bidi: bidi-override }

BDO[DIR=""rtl""] { direction: rtl; unicode-bidi: bidi-override }

\*[DIR=""ltr""] { direction: ltr; unicode-bidi: embed }

\*[DIR=""rtl""] { direction: rtl; unicode-bidi: embed }

";

我通常不会动这个defaultCss，本示例只动了一处，将“small, sub, sup { font-size: .83em }”改成了“small { font-size: .83em }”，否则导出为docx文件后上、下标会偏小。

这三个参数的css会依次覆盖前面的设置，因此authorCss通常应设置为html中外部link的css文件，而写在Html文件<style>标签中的css代码优先级通常更高，因此把它设置为userCss。

本例中，提取了Html中<style>标签中的内容作为authorCss，这个css将**标题1**的背景颜色设置为红色，然后在userCss中将**标题1**的背景颜色设置为绿色，导出后你会发现最终的颜色为绿色。

## 第5步 将OMML公式插入到docx文件的相应位置

代码如下：

// 将OMML公式插入到转换好的文件中

using (MemoryStream memoryStream = new MemoryStream())

{

 doc.WriteByteArray(memoryStream);

 using (WordprocessingDocument wordDoc = WordprocessingDocument.Open(memoryStream, true))

 {

 List<Text> replaceTexts = new List<Text>();

 // 找到<w:b>////</w:b>，这也是我们要插入公式的地方。

 // 此处用的是OpenXML的扩展方法，你也可以使用LINQ To XML完成同样的任务

 foreach (var paragraph in wordDoc.MainDocumentPart.Document.Body.Descendants<Paragraph>())

 {

 foreach (var run in paragraph.Descendants<Run>())

 {

 Text text = run.Elements<Text>().Where(p => p.Text == "////").FirstOrDefault();

 if (text != null)

 replaceTexts.Add(text);

 }

 }

 if (replaceTexts.Count() > 0)

 {

 int i = 0;

 foreach (var replaceText in replaceTexts)

 {

 foreach (var currentRun in ommls[i].Descendants<DocumentFormat.OpenXml.Math.Run>())

 {

 // 添加字体信息

 DocumentFormat.OpenXml.Wordprocessing.RunProperties runProperties2 =

 new DocumentFormat.OpenXml.Wordprocessing.RunProperties();

 RunFonts runFonts2 = new RunFonts() { Ascii = "Cambria Math", HighAnsi = "Cambria Math" };

 runProperties2.Append(runFonts2);

 currentRun.InsertAt(runProperties2, 0);

 }

 replaceText.Parent.ReplaceChild(ommls[i], replaceText);

 i += 1;

 }

 }

 }

 File.WriteAllBytes("../../finalWord.docx", memoryStream.ToArray());

 //wordDoc.SaveAs("../../finalWord.docx");

}

至此就完成了全部工作。

可供下载的源代码是个控制台程序，将一个“Original.html”文件最终转换为“finalWord.docx”。