عنوان: **آشنایی با WebDav و نحوه استفاده از آن**
نویسنده: رضا بازرگان
تاریخ: ۱۳۹۳/۰۹/۱۳ ۱۸:۴۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, MVC, MS Office, webdav, cloud

WebDAV استانداردی است بر روی پروتکل HTTP که Requestها و Responseهای مدیریت یک فایل را بر روی سرویس دهنده وب، تشریح می‌کند.

برای درک چرایی وجود این استاندارد بهتر است ذهن خود را معطوف به نحوه‌ی عملکرد سیستم فایل در OS کنیم که شامل APIهای خاص برای دسترسی نرم افزارهای گوناگون به فایل‌های روی یک سیستم است.
حال فکر کنید یک سرور Cloud راه اندازی نموده‌اید که قرار است مدیریت فایل‌ها و پرونده‌های Office را بر عهده داشته باشد و چون امکان ویرایش اسناد Office بر روی وب را ندارید، نیاز است تا اجازه دهید نرم افزارهای Office مستقیما فایل‌ها را از روی سرور شما باز کنند و بعد از تغییرات، به جای ذخیره در سیستم local، محتوا را به فایل روی سرور ارسال کنند.
در مفهوم web عملا این کار غیر استاندارد و نادرست است. همه درخواست‌ها و جواب‌ها باید بر روی پروتکل Http باشند. خوب حال تصور کنید نرم افزارهای Office قابلیت آن را داشته باشند که به جای تحویل محتوا به سیستم عامل برای ذخیره‌ی آن بر روی سیستم local، محتوا را به یک آدرس ارسال نمایند و پشت آن آدرس، متدی باشد که بتواند به درخواست رسیده، به درستی پاسخ دهد.
این یعنی باید سمت سرور متدی با قابلیت ارسال پاسخ‌های درست و در سمت کلاینت نرم افزاری با قابلیت ارسال درخواست‌های مناسب وجود داشته باشد.
WebDAV استاندارد تشریح محتوای درخواست‌ها و پاسخ‌های مربوط به مدیریت فایل‌ها است.
خوشبختانه نرم افزارهای Office و بسیاری از نرم افزارهای دیگر، استاندارد WebDAV را پشتیبانی می‌کنند و فقط لازم است برای سرورتان متدی با قابلیت پشتیبانی از درخواست‌های WebDAV پیاده سازی نمایید و البته متاسفانه کتابخانه‌های سورس باز چندانی برای WebDAV در سرور دات نت وجود ندارد. من ماژولی را برای کار با WebDAV نوشتم و سورسش را در Git قرار دادم. برای این مثال هم از همین کتابخانه استفاده می‌کنم.
ابتدا یک پروژه‌ی وب MVC ایجاد نمایید و پکیج xDav را از nugget نصب کنید.

```
PM> Install-Package xDav
```

اگر به web.config نگاهی بیاندازیم می‌بینیم یک module به نام xDav به وب سرور اضافه شده که بررسی درخواست‌های WebDAV را به عهده دارد.

```
<system.webServer>
        <modules>
            <add name="XDav" type="XDav.XDavModule, XDav" />
        </modules>
  </system.webServer>
```

همچنین یک Section جدید هم به config برای پیکربندی xDav اضافه شده است.

```
<XDavConfig Name="xdav">
    <FileLocation URL="xdav" PathType="Local"></FileLocation>
  </XDavConfig>
```

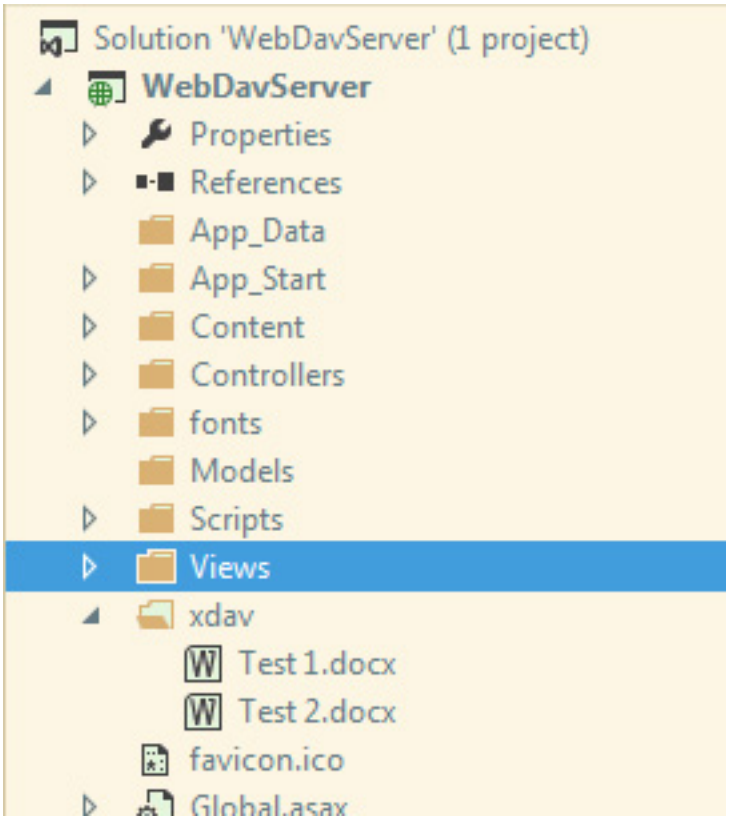
خاصیت Name برای xDav نشانگر درخواست‌هایی است که باید توسط این ماژول اجرا شوند. در اینجا یعنی درخواست‌هایی که آدرس آن‌ها شامل "/xdav/" باشد، توسط این ماژول Handle می‌شوند. عبارت بعد از مقدار Name در URL هم طبیعتا نام فایل مورد نظر شماست.

FileLocation آدرس پوشه ای است که فایل‌ها در آن ذخیره و یا بازخوانی می‌شوند. اگر FileType با مقدار Local تنظیم شود،

یعنی باید یک پوشه به نام خاصیت URL که در اینجا xdav است در پوشه‌ی اصلی وب شما وجود داشته باشد و اگر با Server مقدار دهی شود URL باید یک آدرس فیزیکی بر روی سرور داشته باشد . مثل "c:\webdav"URL=

ما در این مثال مقادیر را به صورت پیش‌فرض نگه می‌داریم. یعنی باید در پوشه‌ی وب، یک Folder با نام xdav ایجاد کنیم.

در ادامه چند فایل word را برای تست در این پوشه کپی می‌کنم.



می خواهیم در صفحه Index، لیستی از فایل‌های درون این پوشه را نمایش دهیم طوری که در صورت کلیک بر روی هر کدام از آن‌ها، آدرس WebDav فایل مورد نظر را به Word ارسال کنیم.

بعد از نصب Office ، در registry چند نوع Url تعریف می‌شود که معرف اپلیکیشنی است که آدرس به آن فرستاده شود. این دقیقا همان چیزیست که ما به آن نیاز داریم. کافیست آدرس WebDav فایل را بعد از عبارت " |ms-word:ofe|u " در یک لینک قرار دهیم تا آدرس به نرم افزار Word ارسال شود. یعنی آدرس URL باید این شکلی باشد:

```
ms-word:ofe|u|http://Webaddress/xdav/filename
```

Webaddress آدرس وبسایت و filename نام فایل مورد نظرمان است. عبارت /xdav/ هم که نشان می‌دهد درخواست هایی که این آدرس را دارند باید توسط ماژول xDav پردازش شوند.

کلاسی با نام DavFile در پوشه‌ی Model ایجاد می‌کنم:

```
public class DavFile
    {
        public string Name { get; set; }
```

```
        public string Href(string webAddress)
        {
            return string.Format("ms-word:ofe|u|http://{0}/xdav/{1}", webAddress, Name);
        }
    }
```

اکشن متد Index را در Home Controller، مانند زیر تغییر دهید:

```
var dir = new DirectoryInfo(XDav.Config.ConfigManager.DavPath);
            var model = dir.GetFiles().ToList()
                                      .Select(f =>
                                          new DavFile() {
                                            Name = f.Name
                                      });
            return View(model);
```

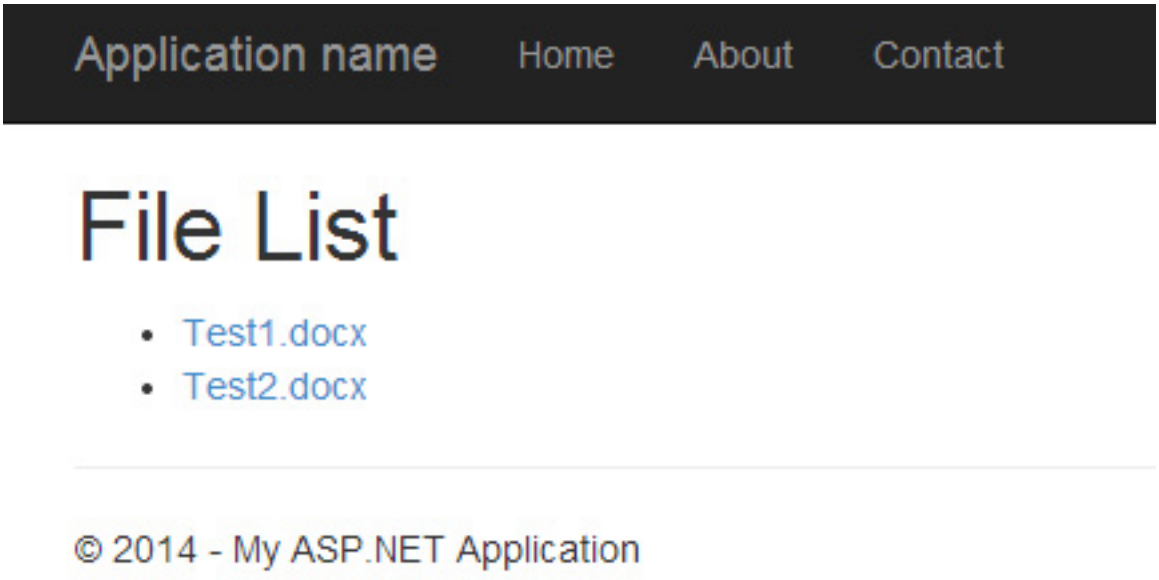
یک لیست از فایل هایی که در پوشه‌ی webDav قرار دارند تهیه می‌کنیم و به View ارسال می‌کنیم. View را هم مثل زیر بازنویسی می‌کنیم.

```
@model IEnumerable<WebDavServer.Models.DavFile>
<h1>
    File List
</h1>

<ul>
    @foreach (var item in Model)
    {
        <li> <a href="@Html.Raw(item.Href(ViewContext.HttpContext.Request.Url.Authority))">
@Html.Raw(item.Name) </a></li>
    }
</ul>
```

قرار است به ازای هر فایل، لینکی نمایش داده شود که با کلیک بر روی آن، آدرس فایل به word ارسال می‌شود. بعد از ثبت تغییرات، word محتوا را به همان آدرس ارسال می‌کند و ماژول xDav محتوا را در فایل فیزیکی سرور ذخیره خواهد کرد.

برنامه را اجرا کنید و بر روی فایل‌ها کلیک نمایید. اگر نرم افزار Office روی کامپیوترتان باز باشد با کلیک بر روی هر کدام از فایل‌ها، فایل word باز شده و می‌توانید محتوا را تغییر داده و ذخیره نمایید.



نرم افزار کلاینت (word) درخواست هایی با verbهای مشخص که در استاندارد WebDav ذکر شده به آدرس مورد نظر می‌فرستد. سرور WebDav درخواست را بر اساس Verb آن Request پردازش کرده و Response استاندارد را ایجاد میکند.

نرم افزار word پس از دریافت یک URL، به جای فرمت فیزیکی فایل، درخواست هایی را با تایپ‌های Option, Head, lock, get, post و unlock ارسال می‌کند. محتوای درخواست و پاسخ هر کدام از تایپ‌ها در استاندارد webDav تعریف شده و ماژول xDav آن را پیاده سازی نموده است.

دریافت پروژه مثال

نظرات خوانندگان

نویسنده: نمو
تاریخ: ۱۳۹۳/۰۹/۱۵ ۱۱:۴۹

سلام؛ ممنون.
آیا از مرورگر خاصی باید استفاده شود؟ پروژه نمونه هم کار نمی‌کند و وقتی روی لینک‌های کلیک می‌کنم هیچ اتفاقی نمی‌افتد.

نویسنده: رضا بازرگان
تاریخ: ۱۳۹۳/۰۹/۱۵ ۱۳:۴۶

با سلام.
مطمئن شوید که Office رو سیستم شما نصب است. من Packageها رو از پروژه نمونه حذف کردم . لطفا مجددا آن را نصب کنید.
فرقی در استفاده از مرورگر هم نیست. می‌توانید از پروژه ای که روی Git گذاشتم هم استفاده کنید که کاملتر است.

عنوان: **استفاده از Interop.word برای جایگزین کردن مقادیر در تمامی فایل (Header - Footer - ...)**
نویسنده: امیر عزیزخانی
تاریخ: ۱۳۹۳/۱۰/۲۰ ۱۹:۱۰
آدرس: www.dotnettips.info
گروه‌ها: C#, PDF, MS Office, MS Word

یکی از متداول‌ترین کارهایی که با اسناد می‌توان انجام داد، تهیه خروجی pdf از word و پر کردن یک فایل word با مقادیر ورودی است که سعی داریم یک نمونه از آن را اینجا بررسی کنیم. کد عمومی برای جایگزین کردن:

```
public void MsInteropReplace(Microsoft.Office.Interop.Word.Application doc, object findText, object
replaceWithText)
        {
            object matchCase = false;
            object matchWholeWord = true;
            object matchWildCards = false;
            object matchSoundsLike = false;
            object matchAllWordForms = false;
            object forward = true;
            object format = false;
            object matchKashida = false;
            object matchDiacritics = false;
            object matchAlefHamza = false;
            object matchControl = false;
            object read_only = false;
            object visible = true;
            object replace = 2;
            object wrap = 1;
            //execute find and replace
            doc.Selection.Find.Execute(ref findText, ref matchCase, ref matchWholeWord,
                ref matchWildCards, ref matchSoundsLike, ref matchAllWordForms, ref forward, ref wrap,
ref format, ref replaceWithText, ref replace,
                ref matchKashida, ref matchDiacritics, ref matchAlefHamza, ref matchControl);
        }
```

و یا این مورد:

```
private static void MsInteropReplace2()
        {
            var doc = new
Microsoft.Office.Interop.Word.Application().Documents.Open(@"D:\temp\te1.docx");

            doc.Content.Find.Execute("@levelOrder", false, true, false, false, false, true, 1, false,
"12345", 2,
            false, false, false, false);
            object missing = System.Reflection.Missing.Value;
            doc.SaveAs(@"D:\temp\out.docx", 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, ref missing);
}
```

که هر دو مورد را در stackoverflow میتوانید پیدا کنید. به شخصه از این مورد برای replace کردن مقادیر در یک فایل template.docx استفاده میکردم؛ ولی بعد از مدتی فهمیدم که Footerها و Header را نمیتواند پردازش کند. کد زیر در تمامی قسمت‌هایی که در یک فایل word می‌توان متغیر تعریف کرد را گشته و عمل پر کردن مقادیر را بر روی فایل نمونه، انجام می‌دهد و شامل سه متد ذیل است:

```
private static void repAll()
        {
            object Missing = System.Reflection.Missing.Value;

            Application app = null;
            Microsoft.Office.Interop.Word.Document doc = null;
            try
            {
                app = new Microsoft.Office.Interop.Word.Application();

                doc = app.Documents.Open(@"D:\temp\te1.docx", Missing, Missing, Missing, Missing,
Missing, Missing, Missing, Missing, Missing);

                FindReplaceAnywhere(app, "@levelOrder", "محرمانه");
```

```
                doc.SaveAs(@"D:\temp\out.docx", Missing, Missing, Missing, Missing, Missing, Missing,
Missing, Missing, Missing, Missing);
            }
            finally
            {
                try
                {
                    if (doc != null) ((Microsoft.Office.Interop.Word._Document)doc).Close(true,
Missing, Missing);
                }
                finally { }
                if (app != null) ((Microsoft.Office.Interop.Word._Application)app).Quit(true, Missing,
Missing);
            }
        }

        private static void searchAndReplaceInStory(Microsoft.Office.Interop.Word.Range rngStory,
string strSearch, string strReplace)
        {
            rngStory.Find.ClearFormatting();
            rngStory.Find.Replacement.ClearFormatting();
            rngStory.Find.Text = strSearch;
            rngStory.Find.Replacement.Text = strReplace;
            rngStory.Find.Wrap = WdFindWrap.wdFindContinue;
            object Missing = System.Reflection.Missing.Value;

            object arg1 = Missing; // Find Pattern
            object arg2 = Missing; //MatchCase
            object arg3 = Missing; //MatchWholeWord
            object arg4 = Missing; //MatchWildcards
            object arg5 = Missing; //MatchSoundsLike
            object arg6 = Missing; //MatchAllWordForms
            object arg7 = Missing; //Forward
            object arg8 = Missing; //Wrap
            object arg9 = Missing; //Format
            object arg10 = Missing; //ReplaceWith
            object arg11 = WdReplace.wdReplaceAll; //Replace
            object arg12 = Missing; //MatchKashida
            object arg13 = Missing; //MatchDiacritics
            object arg14 = Missing; //MatchAlefHamza
            object arg15 = Missing; //MatchControl

            rngStory.Find.Execute(ref arg1, ref arg2, ref arg3, ref arg4, ref arg5, ref arg6, ref arg7,
ref arg8, ref arg9, ref arg10, ref arg11, ref arg12, ref arg13, ref arg14, ref arg15);
        }

        // Main routine to find text and replace it,
        //   var app = new Microsoft.Office.Interop.Word.Application();
        public static void FindReplaceAnywhere(Microsoft.Office.Interop.Word.Application app, string
findText, string replaceText)
        {
            // http://forums.asp.net/p/1501791/3739871.aspx
            var doc = app.ActiveDocument;

            // Fix the skipped blank Header/Footer problem
            //    http://msdn.microsoft.com/en-us/library/aa211923(office.11).aspx
            Microsoft.Office.Interop.Word.WdStoryType lngJunk =
doc.Sections[1].Headers[WdHeaderFooterIndex.wdHeaderFooterPrimary].Range.StoryType;

            // Iterate through all story types in the current document
            foreach (Microsoft.Office.Interop.Word.Range rngStory in doc.StoryRanges)
            {

                // Iterate through all linked stories
                var internalRangeStory = rngStory;

                do
                {
                    searchAndReplaceInStory(internalRangeStory, findText, replaceText);

                    try
                    {
                        //   6 , 7 , 8 , 9 , 10 , 11 -- http://msdn.microsoft.com/en-
us/library/aa211923(office.11).aspx
                        switch (internalRangeStory.StoryType)
                        {
                            case Microsoft.Office.Interop.Word.WdStoryType.wdEvenPagesHeaderStory: // 6
                            case Microsoft.Office.Interop.Word.WdStoryType.wdPrimaryHeaderStory:   // 7
                            case Microsoft.Office.Interop.Word.WdStoryType.wdEvenPagesFooterStory: // 8
                            case Microsoft.Office.Interop.Word.WdStoryType.wdPrimaryFooterStory:   // 9
                            case Microsoft.Office.Interop.Word.WdStoryType.wdFirstPageHeaderStory: //
```

```
10
                            case Microsoft.Office.Interop.Word.WdStoryType.wdFirstPageFooterStory: //
11

                                if (internalRangeStory.ShapeRange.Count > 0)
                                {
                                    foreach (Microsoft.Office.Interop.Word.Shape oShp in
internalRangeStory.ShapeRange)
                                    {
                                        if (oShp.TextFrame.HasText != 0)
                                        {
                                            searchAndReplaceInStory(oShp.TextFrame.TextRange, findText,
replaceText);
                                        }
                                    }
                                }
                                break;

                            default:
                                break;
                        }
                    }
                    catch
                    {
                        // On Error Resume Next
                    }

                    // ON ERROR GOTO 0 -- http://www.harding.edu/fmccown/vbnet_csharp_comparison.html

                    // Get next linked story (if any)
                    internalRangeStory = internalRangeStory.NextStoryRange;
                } while (internalRangeStory != null); //
http://www.harding.edu/fmccown/vbnet_csharp_comparison.html
            }

        }
```

برای تهیه pdf نیز می‌توانید به کد زیر مراجعه کنید:

```
public static void getFileDocxInPdf()
        {
            // Create a new Microsoft Word application object
            Microsoft.Office.Interop.Word.Application word = new
Microsoft.Office.Interop.Word.Application();

            // C# doesn't have optional arguments so we'll need a dummy value
            object oMissing = System.Reflection.Missing.Value;

            // Get list of Word files in specified directory
            DirectoryInfo dirInfo = new DirectoryInfo(@"D:\temp");
            FileInfo[] wordFiles = dirInfo.GetFiles("*.docx");

            word.Visible = false;
            word.ScreenUpdating = false;

            foreach (FileInfo wordFile in wordFiles)
            {
                // Cast as Object for word Open method
                Object filename = (Object)wordFile.FullName;

                // Use the dummy value as a placeholder for optional arguments
                Microsoft.Office.Interop.Word.Document doc = word.Documents.Open(ref filename, ref
oMissing,
                    ref oMissing, ref oMissing, ref oMissing, ref oMissing, ref oMissing,
                    ref oMissing, ref oMissing, ref oMissing, ref oMissing, ref oMissing,
                    ref oMissing, ref oMissing, ref oMissing, ref oMissing);
                doc.Activate();

                object outputFileName = wordFile.FullName.Replace(".docx", ".pdf");
                object fileFormat = WdSaveFormat.wdFormatPDF;

                // Save document into PDF Format
                doc.SaveAs(ref outputFileName,
                    ref fileFormat, ref oMissing, ref oMissing,
                    ref oMissing, ref oMissing, ref oMissing, ref oMissing,
                    ref oMissing, ref oMissing, ref oMissing, ref oMissing,
                    ref oMissing, ref oMissing, ref oMissing, ref oMissing);
```

```
                // Close the Word document, but leave the Word application open.
                // doc has to be cast to type _Document so that it will find the
                // correct Close method.
                object saveChanges = WdSaveOptions.wdDoNotSaveChanges;
                ((_Document)doc).Close(ref saveChanges, ref oMissing, ref oMissing);
                doc = null;
            }

            // word has to be cast to type _Application so that it will find
            // the correct Quit method.
            ((_Application)word).Quit(ref oMissing, ref oMissing, ref oMissing);
            word = null;
        }
```

نظرات خوانندگان

نویسنده: امیر نوروزیان
تاریخ: ۱۳۹۳/۱۰/۲۱ ۸:۳۸

سلام؛ من ازهمین روش شما استفاده کردم چند وقت پیش به وسیله bookmark:

```
private Document oDoc;
        public void createdoc1()
        {
            var realpath="~/template";
            var filePath = Path.Combine(HttpContext.Current.Server.MapPath("~/template"),
Lcourseid.Text + ".doc");
            var oWordApplication = new Application();
            DirectoryInfo dir = new DirectoryInfo(Server.MapPath(realpath));
            foreach (FileInfo files in dir.GetFiles())
            {
                files.Delete();
            }
            // To invoke MyMethod with the default argument value, pass
            // Missing.Value for the optional parameter.
            object missing = System.Reflection.Missing.Value;

            //object fileName = ConfigurationManager.AppSettings["DocxPath"];@"C:\DocXExample.docx";
            string fileName = @"D:\template1.dot";
            //string fileName1 = @"D:\sss.doc";
            object newTemplate = false;

 object docType = 0;
            object isVisible = true;

            //System.Reflection.Missing.Value is used here for telling that method to use default
parameter values when method execution
            oDoc = oWordApplication.Documents.Open(fileName, newTemplate, docType, isVisible, ref
missing, ref missing, ref missing, ref missing, ref missing, ref missing,
                ref missing, ref missing, ref missing, ref missing, ref missing, ref missing);
           // usable in earlier versions of Microsoft Word v2003  v11
          //  if(Convert.ToInt16(oWordApplication.Version) >=11)
            {
             //Sets or returns a Boolean that represents whether a document is being viewed in reading
layout view.
              oDoc.ActiveWindow.View.ReadingLayout = false;
           }

            //The active window is the window that currently has focus.If there are no windows open, an
exception is thrown.
            //microsoft.office.tools.word.
            oDoc.Activate();

            if (oDoc.Bookmarks.Exists("Title"))
            {
                oDoc.Bookmarks["Title"].Range.Text = "Test Field Entry from webform";
                oDoc.Bookmarks["Address"].Range.Text = "Address Field Entry from webform";
            }

            oDoc.SaveAs(filePath, ref missing);
            oWordApplication.Documents.Close(ref missing, ref missing, ref missing);
            //oWordApplication.Quit(ref SaveChanges, ref missing, ref missing, ref missing);
            ProcessRequest(filePath, Lcourseid.Text);
```

ولی این روش مشکلاتی هم داره. اول اینکه باید روی سرور تنظیمات خاصی رو انجام بدی. البته از تنظیمات منظور تنظیمات دسترسی کاربران هست. ولی استفاده از داک ایکس بیشتر استقبال میشه چون دردسرش کمتره.

نویسنده: امیر عزیزخانی
تاریخ: ۱۳۹۳/۱۰/۲۱ ۲۰:۶

البته Docxابزار خوبیه ولی توی یک مثالی که خواستم ازش استفاده کنم به EXCEPTION خورد و علتش تا اونجایی که من متوجه شدم گیر دادن به محتویات اضافه XML داخل فایل بود