

# Creating a web part to expose an Open Xml Document metadata in Windows Sharepoint Services 3.0

## Prerequisites

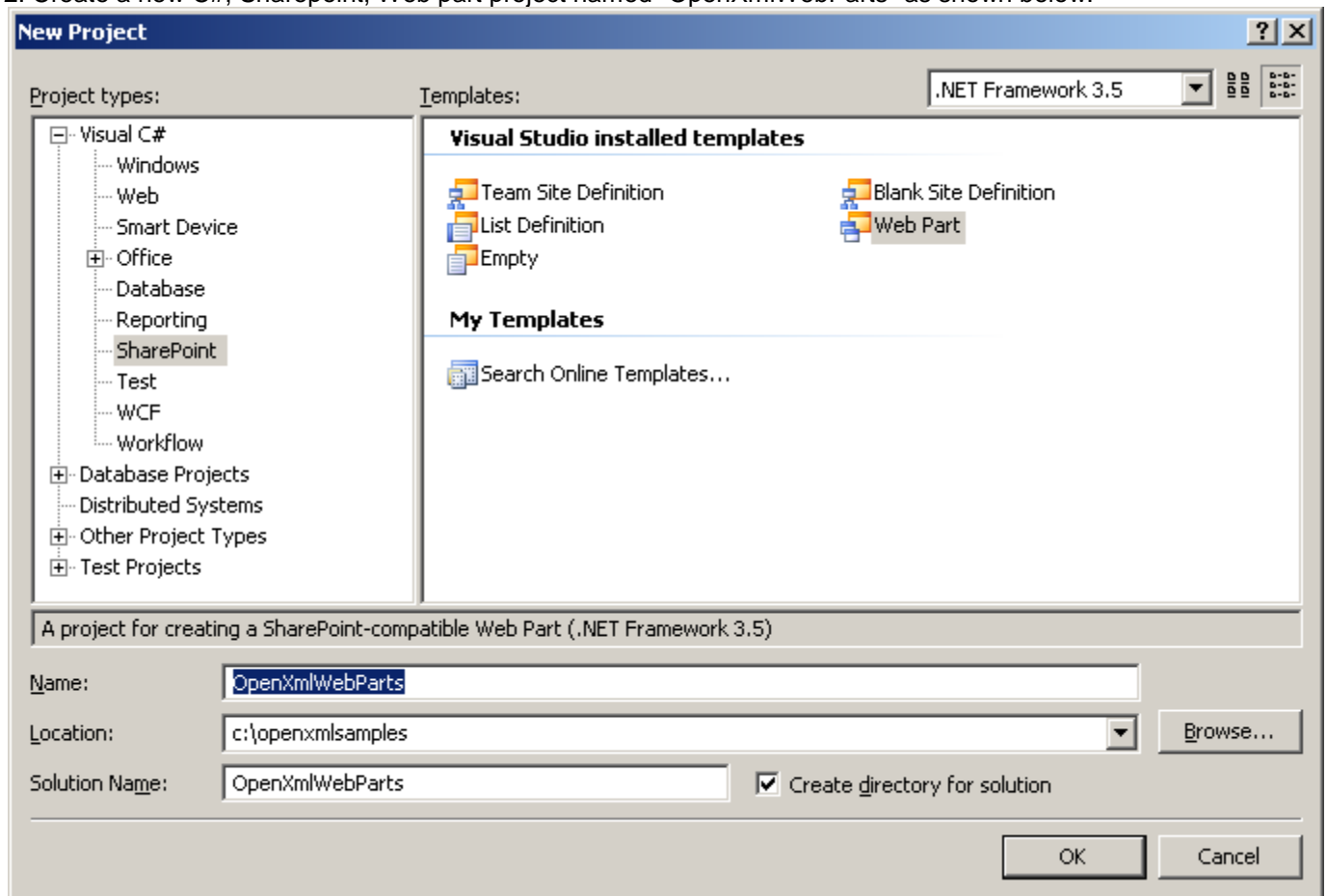
The following software must be already installed in your machine:

1. Microsoft Visual Studio 2008
2. [Windows Sharepoint Services 3.0 Extensions for Visual Studio 2008](#)
3. [Microsoft Open Xml SDK v2.0 \(April 2009 CTP\)](#)

## Step by Step

### Creating the web part

1. Open Visual Studio 2008
2. Create a new C#, Sharepoint, Web part project named "OpenXmlWebParts" as shown below:



3. Add the following references to the solution

- System.Core
- System.Data
- System.Xml.Linq
- DocumentFormat (browse to the following file "C:\Program Files\Open XML Format SDK\V2.0\lib\DocumentFormat.OpenXml.dll")

4. Rename the WebPart1 folder as OpenXmlMetaDataWebPart

5. Rename the files WebPart1.cs, WebPart1.webpart and WebPart1.xml as

OpenXmlMetaDataWebPart.cs, OpenXmlMetaDataWebPart.webpart and OpenXmlMetaDataWebPart.xml respectively

6. Open the code view for OpenXmlMetaDataWebPart.cs

7. Rename the webpart as OpenXmlMetaDataWebPart (remove the default constructor)

8. Because we are going to use this webpart in WSS then we need to change the base class of our web part from `System.Web.UI.WebControls.WebParts.WebPart` to

`Microsoft.SharePoint.WebPartPages.WebPart`

9. We are not going to use the `CreateChildContentControls()` method so we can remove it from the web part class

10. Add the following private member to the web part class (this member is going to be used to make the connection with the source document library)

```
/// <summary>
    /// Private member to make the connection with the source
code Library
    /// </summary>
    private IWebPartRow _sourceDocumentLibrary;
```

11. Add the following private member to the web part class in order to store the information for the selected row at the document library

```
/// <summary>
    /// Private member to store the information for the selected
row at the document library
    /// </summary>
    private DataRowView _tableData;
```

12. Right click on the `DataRowView` class name in the previous line of code and choose the `Resolve` option from the context menu in order to include the `System.Data` namespace reference at the beginning of the class code file

13. Add the following method in order to make this web part aware of web parts implementing the `IWebPartRow` interface at the same `WebPartZone` (the Document Libraries at WSS implement that interface)

```
/// <summary>
    /// Set the data row web part provider
    /// </summary>
    /// <param name="provider"></param>
    [ConnectionConsumer("Row")]
    public void SetConnectionInterface(IWebPartRow provider)
    {
        _sourceDocumentLibrary = provider;
    }
```

14. Add the following method to the web part class (this is the callback method which is going to be used to get the data for the selected row at the document library):

```
/// <summary>
    /// Gets the data for the selected row at the document library
    /// </summary>
    /// <param name="rowData"></param>
    private void GetRowData(object rowData)
    {
        _tableData = (DataRowView) rowData;
    }
```

15. Add the following method to the web part in order to use the callback method defined previously to get the data for the selected row at the document library (we are doing this previously to render the document metadata):

```

protected override void OnPreRender(EventArgs e)
{
    if (_sourceDocumentLibrary != null)
    {
        _sourceDocumentLibrary.GetRowData(new
RowCallback(GetRowData));
    }
}

```

16. Add the following method to the webpart in order to render the document metadata:

```

protected override void RenderContents(HtmlTextWriter writer)
{
    if (_sourceDocumentLibrary != null)
    {
        if (_tableData != null)
        {
            //Get the web site where this web part is running
            SPWeb web = SPControl.GetContextWeb(Context);
            //Get the file we want to show the metadata for
            //We have to concatenate the web url with the
file url
            //The file url is located at the "DocUrl" column
for the DataRowView receive from the Document Library
            SPFile file = web.GetFile(web.Url +
_tableData.Row["DocUrl"].ToString());

            //Get the Document Properties

System.Collections.Generic.Dictionary<string, string>
docProperties
                =
OpenXmlDocumentManager.GetDocumentProperties(file);

            //Render the document properties as a table in
the webpart

writer.RenderBeginTag(HtmlTextWriterTag.Table);

            foreach (string key in docProperties.Keys)
            {

writer.RenderBeginTag(HtmlTextWriterTag.Tr);

writer.RenderBeginTag(HtmlTextWriterTag.Td);
                writer.Write(key);
                writer.RenderEndTag();

writer.RenderBeginTag(HtmlTextWriterTag.Td);

```

```

        writer.Write(docProperties[key]);
        writer.RenderEndTag();
        writer.RenderEndTag();
    }

    writer.RenderEndTag();
}
else
{
    writer.Write("No document selected");
}
}
else
{
    writer.Write("Not connected");
}
}
}

```

17. We are done with the webpart. Now we are going to add the necessary classes to extract the metadata information for the selected Open Xml Document at the Document Library. We are going to use here portions of code extracted from [this really nice](#) post by Eric White

18. Add a new folder to the web part project named "OpenXmlClasses"

19. Add a new class inside the "OpenXmlClasses" folder named OpenXmlExtensions (this class will contain an extension method to extract the content for an Open Xml Document Part)

20. Make the OpenXmlExtensions class public and static

20. Add the following method to the OpenXmlExtensions class

```

public static XDocument GetXDocument(this OpenXmlPart part)
{
    XDocument xdoc = part.Annotation<XDocument>();

    if (xdoc != null)

        return xdoc;

    using (StreamReader sr = new
StreamReader(part.GetStream()))

        using (XmlReader xr = XmlReader.Create(sr))

            xdoc = XDocument.Load(xr);

    part.AddAnnotation(xdoc);

    return xdoc;
}

```

21 Use the technique shown in step 12 in order to resolve the namespaces for the OpenXmlPart, XDocument, StreamReader and XmlReader classes referenced at the previously created method

22. Add a new class inside the OpenXmlClasses folder named "OpenXmlDocumentManager"

23. Make the OpenXmlDocumentManager class public

24 Add the following method the OpenXmlDocumentManager class (this method will return the document properties extracted from the ExtendPropertiesPart of the WordProcessingML document as a Dictionary):

```
/// <summary>
    /// Returns the extended properties for a WordprocessingML
document stored in a WSS web site
    /// </summary>
    /// <param name="file">Reference to the file stored in a WSS
web site</param>
    /// <returns>Dictionary containing the extended properties
for the WordProcessingML document</returns>
    public static Dictionary<string, string>
GetDocumentProperties(SPFile file)
    {
        byte[] byteArray = file.OpenBinary();
        Dictionary<string, string> documentProperties = new
Dictionary<string, string>();

        using (MemoryStream mem = new MemoryStream())
        {
            mem.Write(byteArray, 0, (int)byteArray.Length);
            try
            {
                using (WordprocessingDocument wordDoc =
                    WordprocessingDocument.Open(mem, true))
                {
                    ///Getting the ExtendedFileProperties part
for the document
                    ExtendedFilePropertiesPart fileProperties
= wordDoc.ExtendedFilePropertiesPart;
                    XDocument filePropertiesXDoc =
fileProperties.GetXDocument();

                    //Adding the properties to the dictionary
                    //In this sample we are only returning the
pages, paragraphs and characters extended properties
                    documentProperties.Add("Pages",
fileProperties.Properties.Pages.Text);
                    documentProperties.Add("Paragraphs",
fileProperties.Properties.Paragraphs.Text);
                    documentProperties.Add("Characters",
fileProperties.Properties.Characters.Text);

                }

                return documentProperties;
            }
        }
    }
}
```

```

        catch (Exception ex)
        {

            return null;

        }

    }
}

```

25. Use the technique shown in step 12 in order to resolve the namespaces for the SPFile, MemoryStream, XDocument and WordprocessingDocument classes referenced in the previously created method
26. Go the Web Part code file view, look for the RenderContents method and use the technique shown in step 12 in order to resolve the namespace for the XmlDocumentManager class referenced there
27. Build the solution

### **Deploying the web part**

1. Go the bin\debug folder for your solution at Windows Explorer
2. Create a new text file named OpenXmlMetadataWebPart.dwp in that folder
3. Open the OpenXmlMetadataWebPart.dwp file created at the previous step (you can use Notepad to do that)
4. Add the following content to that file
 

```

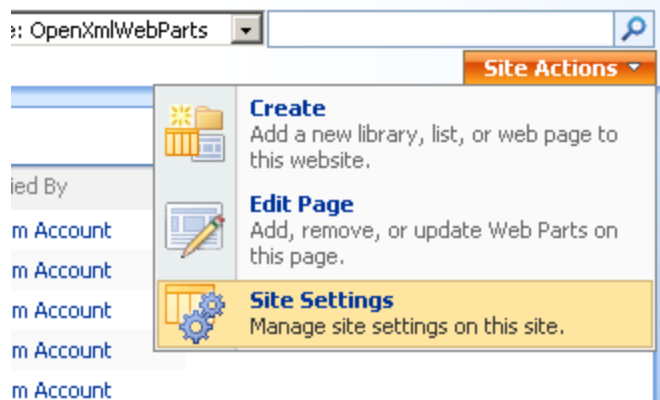
<?xml version="1.0"?>
<WebPart xmlns="http://schemas.microsoft.com/WebPart/v2">
  <Assembly>OpenXmlMetadataWebPart, Version=1.0.0.0, Culture=Neutral,
PublicKeyToken=[CopyHereTheAssemblyPublicKeyToken]</Assembly>
  <TypeName>OpenXmlMetadataWebPart.OpenXmlMetadataWebPart</TypeName>
  <Title>Open Xml Metadata Web Part</Title>
  <Description>Sample Web Part on how to display WordprocessingML document
metadata</Description>
</WebPart>

```
5. Open a Visual Studio 2008 Command Prompt and browse to the bin\debug folder for your solution
6. Once there type the following command: sn -T OpenXmlWebParts.dll
7. The sn utility helps us to extract the Public Key Token for the Web Part assembly. We need this token in order to register the web part at WSS. Replace the "[CopyHereTheAssemblyPublicKeyToken]" entry at the dwp file created at step 4 with the returned token from invoking sn at step 6. Save the .dwp file
8. Copy the web part .dll and .dwp files to the bin folder for the WSS web site you want to use the web part (this folder is usually located at c:\inetpub\wwwroot\wss\virtualdirectories\[websiteportnumber])
9. Open the web.config file for the WSS web site you want to use the web part
10. Look for the <trust> element entry and set the level attribute to "WSS\_Medium"
11. Look for the <SafeControls> entry and add the following element inside it:
 

```

<SafeControl Assembly="OpenXmlMetadataWebPart, Version=1.0.0.0,
Culture=neutral, PublicKeyToken=[CopyHereTheAssemblyPublicKeyToken]"
Namespace="OpenXmlWebParts" TypeName="OpenXmlMetadataWebPart" Safe="True"/>

```
12. Replace the "[CopyHereTheAssemblyPublicKeyToken]" value at previously added entry with the public key token for the web part obtained at step 6
13. Save the web.config file
14. Open the WSS web site where are deploying the web part
15. Click the "Site Actions" option near the right upper corner and then click on "Site Settings"



16. Select the Web Parts option located under the "Galleries" tab

## Site Settings

**Site Information**

Site URL:	http://staff_der
Mobile Site URL:	http://staff_der
Version:	12.0.0.4518

**Users and Permissions**

**Look and Feel**

**Galleries**

- People and groups
- Site collection administrators
- Advanced permissions

- Title, description, and icon
- Tree view
- Site theme
- Top link bar
- Quick Launch
- Save site as template
- Reset to site definition

- Master pages
- Site content types
- Site columns
- Site templates
- List templates
- Web Parts
- Workflows

17. Select the "Upload" option

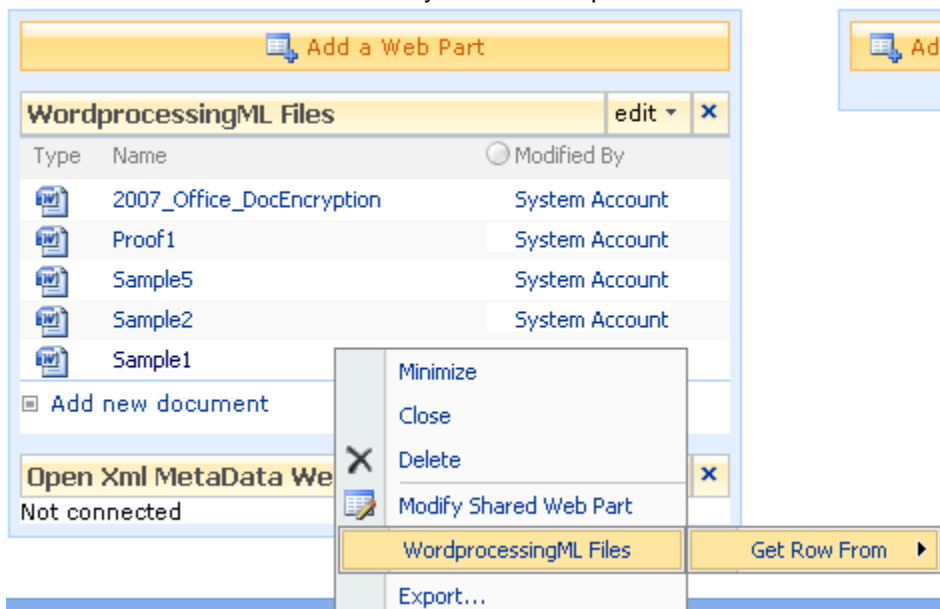


18. Browse for the OpenXmlMetaDataWebPart.dwp file located at the WSS Web Site bin folder and click Ok
19. Change the "Group" and "Quick Add Groups" options are shown below and click Ok
20. We are done. Our web part is already deployed at WSS Web Site

### Using the Open Xml MetaData Web Part






I'm assuming basic knowledge about configuring a page in a WSS Web Site

1. Go the main page of the WSS Web Site you deployed the Open Xml MetaData Web Part
2. Create a new Document Library to add some .docx files you can use to test our web part
3. Add the Document Library created in the previous step to the WSS web site main page (or another page you created for testing purposes)
4. Add a new instance of the Open Xml MetaData Web Part
5. Select the Edit, Connections, "Get Row From" option for the Open Xml MetaData Web Part in order to connect it with the Document Library added at step 3:



6. Click Exit Edit Mode at the upper right corner
7. We are done!!! Select a file at the Document Library and you will see its metadata info displayed in our web part:

### WordprocessingML Files

	Type	Name	Modified By
<input checked="" type="radio"/>		2007_Office_DocEncryption	System Account
<input type="radio"/>		Proof1	System Account
<input type="radio"/>		Sample5	System Account
<input type="radio"/>		Sample2	System Account
<input type="radio"/>		Sample1	System Account

[Add new document](#)

### Open Xml MetaData Web Part

Pages 20  
Paragraphs 48  
Characters 20686