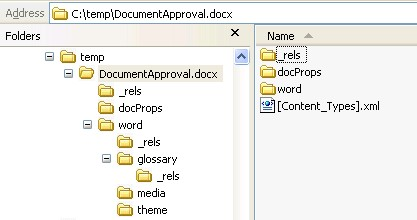
Lab 10: Integration with Office Open XML File Formats

**Lab Time**: 30 Minutes

**Lab Directory**: C:/Student/Labs/10\_OpenXml

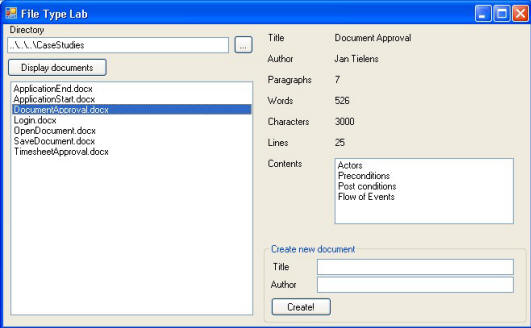
**Lab Overview**: Litware has 100s of Case Studies gathered through the years. Our goal is to move all those documents into SharePoint. But before that, your job is to build a tool that helps administrators discover what we have in our inventory of Case Studies.

# Exercise 1: Examine a DocX file through Windows Explorer



1. Copy the **CaseStudies** folder from the **StarterFiles** folder to the **Lab** folder and open the file **Lab\CaseStudies\DocumentApproval.docx** in Word (you can find it in the **StarterFiles\CaseStudies** directory), by double clicking on it. Take a look at the contents of this file. Notice that there is an image and some headings.
2. Close Word and rename the **DocumentApproval.docx** file to **DocumentApproval.docx.zip** and open it.
3. Navigate to the word directory in the zip file and open the **document.xml** file. Review the content of this **WordML** file and search for the different headings in the XML.
4. Close the **document.xml** file and navigate to the media directory, then open the **image1.jpeg** file. Close the file.
5. Navigate back to the **word** directory and then to the **\_rels** directory and open the **document.xml.rels** file. In this file you’ll find the **rId6** relationship that points to the image you’ve just opened. Close the file.
6. Next open the **document.xml** file in the word directory to find where the image is referenced in the xml. Close the file.
7. Finally go back to the root of the zip file, navigate to the **docProps** directory, and open the **app.xml** where you’ll find all the application specific properties for this document. Close the file.
8. Let’s make a small change to the Word document without actually using the Word application. Extract the contents of the **DocumentApproval.docx.zip** to a folder by right clicking on the zip folder and choosing **Extract All**. Now go to the newly extracted folder (if you accepted the defaults in the extract process, then it will be in the same directory as the **zip** folder) and navigate to the **word** directory.
9. Open the file footer2.xml in Microsoft Visual Studio 2008 and look for the word Confidential, change it and any surrounding text to Litware Confidential 2008 - Don't distribute this document! Save the changes.
10. Now we need to copy **the footer2.xml** file back to the **zip** folder and compress it back into a **docx** file by doing the following:
    1. After you have saved the **footer2.xml** file, go to the directory you saved it into (the folder that you extracted from the zip file).
    2. Right click on **footer2.xml** and click **Copy**.
    3. Navigate to **DocumentApproval.docx.zip > word** and paste the **footer2.xml** file there—you will have to approve that you want to replace the existing file.
    4. Navigate back to the **Case Studies** folder.
    5. Delete the extracted folder called **DocumentApproval.docx**.
    6. Change the name of DocumentApproval.docx.zip back to DocumentApproval.docx.
    7. Verify the changes by opening the document in Word.

# Exercise 2: Build the Case Study explorer tool



1. Copy the **FileTypeLab** folder from the **StarterFiles** into the **Lab** Folder.
2. Open **Visual Studio 2008**, and go to **File>Open>Project/Solution**, navigate to the folder you just copied, and open the **FileTypeLab.sln** file.
3. Start the application (press **F5**) to check out what’s already there. When you click the ellipsis button […] to the right of the text box where the directory name is displayed, you can select a directory that will be processed.
4. Read the following sub-point for understanding. Note: you are not being asked to complete a task. This is just for explanation:
5. Once you’ve selected a directory you can click the **Display documents** folder that will load all files with the **.docx** extension in the list. Selecting a document does not do anything yet. We expect you to write this code. Also the **Create** button at the bottom right of the form isn’t implemented yet. If you have time, this can be done as an optional exercise.
6. Now close the application so we can start writing some code.
7. First, add a reference to **WindowsBase.dll** which contains the new .NET 3.0 packaging API definitions within the **System.IO.Packaging** namespace. Note that you might not see this assembly as one of the component names inside the .NET tab of the **Add References** dialog. If this is the case, you will have to use the Browse tab to navigate to the following directory to add **WindowsBase.dll** as a reference within the current project:

C:\Program Files\Reference Assemblies\Microsoft\Framework\v3.0

1. Next open the code for Form1 and add the following using statements at the top:

using System.IO;

using System.Xml;

using System.IO.Packaging;

1. Open the region named Exercise 2, and add the following code to the **ProcessDocument** method. This code opens the package and receives the file name as a parameter.

Package package = Package.Open(fullFileName, FileMode.Open);

1. The following code loads the core properties out of the package. We need these properties to retrieve the title and the author of the document. Add this code below the code you just added in the **Exercise 2** region:

// Get core properties

XmlDataDocument doc = new XmlDataDocument();

Uri uri = new Uri("/docProps/core.xml", UriKind.Relative);

PackagePart propsPart = package.GetPart(uri);

doc.Load(propsPart.GetStream(FileMode.Open));

XmlNamespaceManager nsmgr = new XmlNamespaceManager(doc.NameTable);

nsmgr.AddNamespace("cp",

"http://schemas.openxmlformats.org/package/2006/metadata/core-properties");

nsmgr.AddNamespace("dc", "http://purl.org/dc/elements/1.1/");

titleTextBox.Text = doc.SelectSingleNode("cp:coreProperties/dc:title", nsmgr).InnerText;

authorTextBox.Text = doc.SelectSingleNode("cp:coreProperties/dc:creator",

nsmgr).InnerText;

1. Retrieving the application specific properties is done in the same fashion. But before we do that, we're going to check if that part is available in the package. Place this code under the last code.

// Get application specific properties

doc = new XmlDataDocument();

uri = new Uri("/docProps/app.xml", UriKind.Relative);

if (package.PartExists(uri))

{

PackagePart appPart = package.GetPart(uri);

doc.Load(appPart.GetStream(FileMode.Open));

nsmgr = new XmlNamespaceManager(doc.NameTable);

nsmgr.AddNamespace("ap", "http://schemas.openxmlformats.org/officeDocument/2006/extended-properties");

paragraphsTextBox.Text = doc.SelectSingleNode("ap:Properties/ap:Paragraphs", nsmgr).InnerText;

wordsTextBox.Text = doc.SelectSingleNode("ap:Properties/ap:Words", nsmgr).InnerText;

characterTextBox.Text = doc.SelectSingleNode("ap:Properties/ap:Characters", nsmgr).InnerText;

linesTextBox.Text = doc.SelectSingleNode("ap:Properties/ap:Lines", nsmgr).InnerText;

}

else

{

paragraphsTextBox.Text = "na";

wordsTextBox.Text = "na";

characterTextBox.Text = "na";

linesTextBox.Text = "na";

}

1. Finally we need to write some code to retrieve the contents of the document. We are going to list all the items that have the Heading2 style applied in the list box. To do so, you can use the WordML available in the document.xml file. Place this code under the last code you added.

// Get contents

contentsListBox.Items.Clear();

doc = new XmlDataDocument();

uri = new Uri("/word/document.xml", UriKind.Relative);

PackagePart docPart = package.GetPart(uri);

doc.Load(docPart.GetStream(FileMode.Open));

nsmgr = new XmlNamespaceManager(doc.NameTable);

nsmgr.AddNamespace("w","http://schemas.openxmlformats.org/wordprocessingml/2006/3/main");

foreach (XmlNode node in doc.SelectNodes("//\*[name()='w:pStyle']", nsmgr))

{

if (node.Attributes["w:val"].Value == "Heading2")

{

System.Xml.XPath.XPathNavigator nav = node.CreateNavigator();

nav.MoveToParent();

nav.MoveToNext();

nav.MoveToChild("t",

“http://schemas.openxmlformats.org/wordprocessingml/2006/3/main");

contentsListBox.Items.Add(nav.Value);

}

}

1. To conclude, we need to close to package opened.

// Close the package

package.Close();

1. Press **F5** to start your application and verify your work. You can find some sample Case Studies in the folder **Lab\CaseStudies**. Close the application before moving on to **Exercise 3**.

# Optional Exercise 3: Extend to explorer to create new documents using The Open XML Format SDK

1. Add a reference to the **DocumentFormat.OpenXml** assembly.
2. Add the necessary namespace to work with the **Open Xml Format SDK**:

using DocumentFormat.OpenXml;

using DocumentFormat.OpenXml.Packaging;

1. Open the region **Exercise 3** and add the following code to the **CreateNewDocument** method. Use the **Create** method of the **WordprocessingDocument** class. This code creates a new object of type **WordprocessingDocument**.

// create the package

string fullFileName = directoryTextBox.Text + "\\" + newTitleTextBox.Text + ".docx";

using (WordprocessingDocument wordDoc = WordprocessingDocument.Create(fullFileName,

WordprocessingDocumentType.Document))

{

// add code here

}

1. In the folder **Labs\FileTypeLab** you find a file with the name **part\_main.xml**. It contains the contents for the main part of the word document. Inspect the content.
2. Remove the comment within the using statement and add code to create the main part of the document. Load the **part\_main.xml** into the main document part and save the changes.

// Set the content of the document so that Word can open it.

MainDocumentPart mainPart = wordDoc.AddMainDocumentPart();

StreamWriter partWriter =

new StreamWriter(wordDoc.MainDocumentPart.GetStream(FileMode.Create,

FileAccess.Write));

XmlDocument doc = new XmlDocument();

doc.Load(@"..\..\part\_main.xml");

doc.Save(partWriter);

partWriter.Close();

1. Now you are going to add a document part for the core properties. In the folder **Labs\FileTypeLab** you find a file with the name **part\_core.xml**. It contains the contents for the word document properties. Inspect the content.
2. Then add a document part for the document core properties. Load the contents of **part\_core.xml** into the new part:

// Add properties part

CoreFilePropertiesPart propertiesPart = wordDoc.AddCoreFilePropertiesPart();

StreamWriter propWriter =

new StreamWriter(wordDoc.CoreFilePropertiesPart.GetStream(

FileMode.Create, FileAccess.Write));

// First load the part\_core.xml file

XmlDocument propdoc = new XmlDocument();

propdoc.Load(@"..\..\part\_core.xml");

1. You are going to fill out some of the document properties like the title and the author of the document. You can achieve this by using **XPath**. First create an instance of type **XmlNamespaceManager** and add the necessary namespaces for accessing the package and the core properties:

// Manage namespaces to perform XML XPath queries.

NameTable nt = new NameTable();

XmlNamespaceManager nsmgr = new XmlNamespaceManager(nt);

nsmgr.AddNamespace("cp",

"http://schemas.openxmlformats.org/package/2006/metadata/core-properties");

nsmgr.AddNamespace("dc", "http://purl.org/dc/elements/1.1/");

1. Fill out the properties like **title** and **creator** by using **XPath**:

// Then set our own values

propdoc.SelectSingleNode("cp:coreProperties/dc:title", nsmgr).InnerText =

newTitleTextBox.Text;

propdoc.SelectSingleNode("cp:coreProperties/dc:creator", nsmgr).InnerText =

newAuthorTextBox.Text;

propdoc.Save(propWriter);

propWriter.Close();

1. Notice that you don’t have to add relationships anymore.
2. Application properties like word count, line count, paragraph count are part of the extended document properties. Therefore you have to create a document part of type **ExtendedFileProperties** part:

// Set application specific properties

ExtendedFilePropertiesPart extpropsPart = wordDoc.AddExtendedFilePropertiesPart();

StreamWriter extpropsWriter =

new StreamWriter(extpropsPart.GetStream(FileMode.Create, FileAccess.Write));

XmlDocument extdoc = new XmlDocument();

extdoc.Load(@"..\..\part\_ext.xml");

extdoc.Save(extpropsWriter);

extpropsWriter.Close();

1. Word documents also contain styles. You can add a **StyleDefinitionPart** to the document but this is a child of the **MainDocumentPart** (and not a child of the document like the **CoreFilePropertiesPart**)

// add styles to it

if (wordDoc.MainDocumentPart.StyleDefinitionsPart != null)

{

wordDoc.MainDocumentPart.DeletePart(

wordDoc.MainDocumentPart.StyleDefinitionsPart);

StyleDefinitionsPart stylePart =

wordDoc.MainDocumentPart.AddNewPart<StyleDefinitionsPart>();

StreamWriter styleWriter =

new StreamWriter(stylePart.GetStream(FileMode.Create, FileAccess.Write));

XmlDocument styledoc = new XmlDocument();

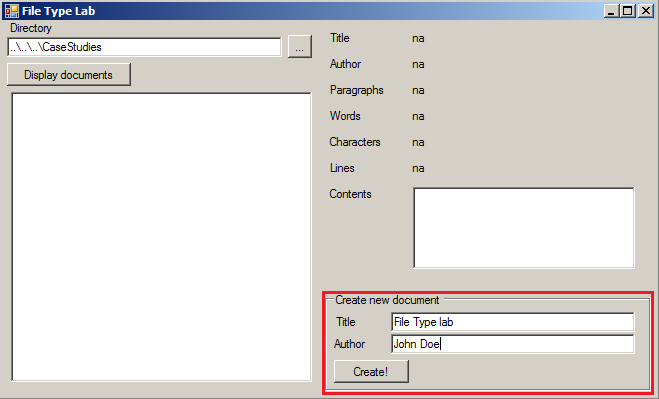
styledoc.Load(@"..\..\part\_styles.xml");

styledoc.Save(styleWriter);

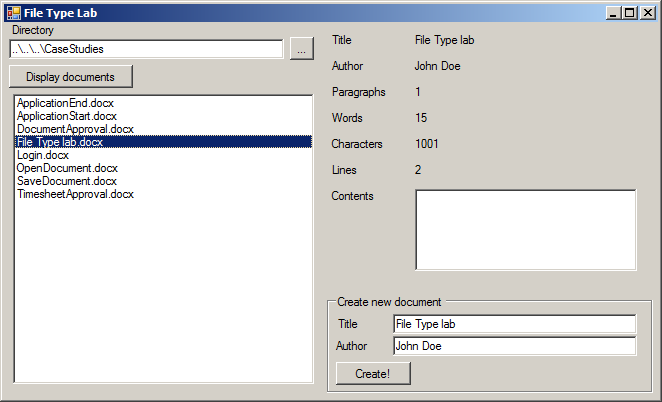
styleWriter.Close();

}

1. This finishes off the exercise. Save and build your work. Pres **F5** to run the application. Fill out a title and your name in the **Create new document** group box. Click the **Create** button to create the document.



1. Click the **Display documents** button again and select your newly created document. The document properties should get displayed.



# Student Challenge: Write file format explorer with Windows tree view

As a challenge, think about creating a small Windows application an administrator can use to view the internals of an Office 2007 document. Use a tree view control to display the container structure in a panel. When the user selects a node representing an XML-based document part or a relationship, display the XML content in a second panel using an embedded Web browser control.