|  |
| --- |
| Microsoft SharePoint 2013 - Hands-on Lab |
| Office Services |
| Verified Against Build #15.0.4420.1017 |

|  |
| --- |
| Microsoft  Version 1.0  August 14, 2012 |

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

The names of manufacturers, products, or URLs are provided for informational purposes only and Microsoft makes no representations and warranties, either expressed, implied, or statutory, regarding these manufacturers or the use of the products with any Microsoft technologies. The inclusion of a manufacturer or product does not imply endorsement of Microsoft of the manufacturer or product. Links are provided to third party sites. Such sites are not under the control of Microsoft and Microsoft is not responsible for the contents of any linked site or any link contained in a linked site, or any changes or updates to such sites. Microsoft is not responsible for webcasting or any other form of transmission received from any linked site. Microsoft is providing these links to you only as a convenience, and the inclusion of any link does not imply endorsement of Microsoft of the site or the products contained therein.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Copyright © 2012 Microsoft Corporation. All rights reserved.

Microsoft and SharePoint are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

# Introduction

## Estimated time to complete this lab

60 minutes

## Objectives

After completing this lab, you will be able to:

* Create features using new capabilities of Excel Services
* Create features using new capabilities of Word Automation Services
* Create features using new capabilities of PowerPoint Services
* Create features using new capabilities of Translation Services

## Overview of Lab

This lab introduces the new capabilities of Office Services. The lab will walk you through creating features that utilize each of the services.

## Virtual Machine Technology

The computers in this lab are virtual machines that are implemented using Microsoft Hyper-V. Before starting each virtual machine, ensure you apply the **Start-Lab** snapshot. When you have started a virtual machine, log on by pressing **CTRL+ALT+END** and supply the credentials listed in the lab instructions.

## Computers in this lab

This lab uses virtual machines as described in the following table. Before you begin the lab, you must start the virtual machines and then log on to the computers.

|  |  |
| --- | --- |
| **Virtual Machine** | **Role** |
| {Supplied by Instructor} | Domain Controller |
| {Supplied by Instructor} | Actual SharePoint environment with Office client and other required software. |

All user accounts in this lab use the password {Supplied by Instructor}.

## Create Lab SharePoint Site Collection

In the files provided with the hands on lab, run the batch file called **SetupModule.bat** by double clicking it. This file will execute a PowerShell script that will create a new site collection at [http://intranet.contoso.com/sites/](http://wave15-sp/sites/IntroSpApps)OfficeServices.

# Exercise 1: Excel Services JavaScript

In this lab, you will investigate the capabilities of Excel Services.

## Task 1 – Upload a Spreadsheet

In this task, you will create a document library and upload a spreadsheet.

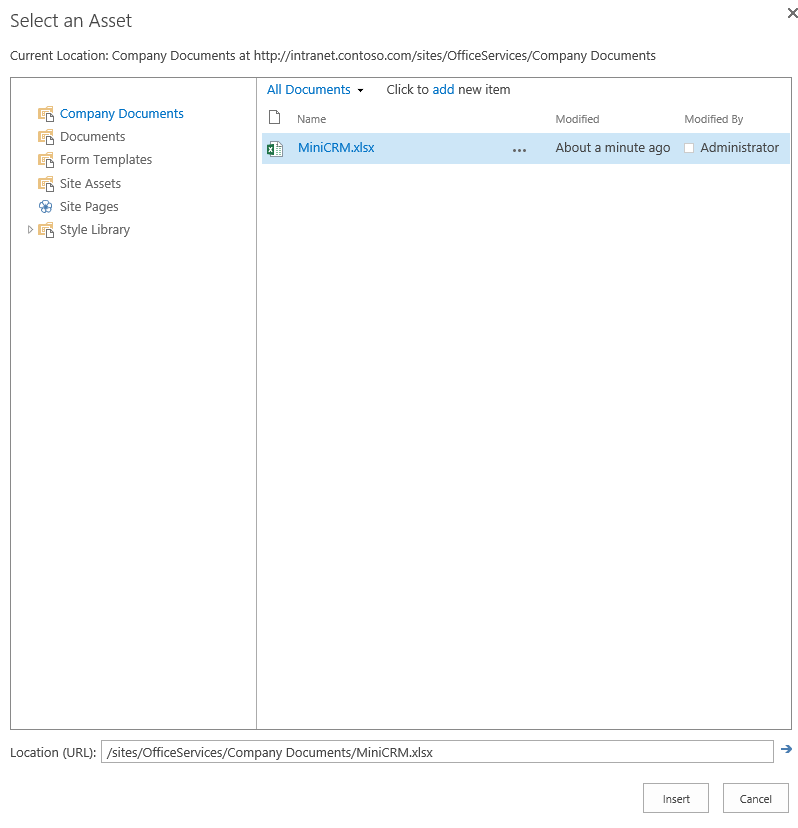
* This task requires the file MiniCRM.xlsx.

1. **Navigate** to the home page of a site where you can work on the lab.
2. Click **Site Contents**.
3. Click **Add an App**.
4. On the Add an App page, click **Document Library**.
5. Name the Document Library **Company Documents** and click **Create**.
6. On the Apps page, click **Company Documents**.
7. **Drag** the file **MiniCRM.xlsx** onto the document library to add the spreadsheet to the Shared Documents library.

## Task2 - Using the Excel Services Web Part

In this task, you will display the spreadsheet using the Excel Services web part.

1. **Navigate** to the home page of the site.
2. **Click** the **Page** tab in the ribbon.
3. **Click** the **Edit Page** button.
4. Click **Add a Web Part**.
5. In the **Business Data** folder, select the **Excel Web Access** web part and click **Add**.
6. In the Excel Web Access web part, click **Click Here to Open the Tool Pane**.
7. In the tool pane, click the **Workbook** ellipsis.
8. In the Select an Asset dialog, select the **MiniCRM.xlsx** file and click **Insert**.



* + - * + Select the MiniCRM.xlsx file

1. In the tool pane, click **OK**.
2. Click **Add a Web Part**.
3. In the **Media and Content** folder, select the **Script Editor** web part and click **Add**.
4. Click **Click to Edit this Snippet**.
5. **Add** the following code to the **Embed** dialog and click **Insert**.

<script type="text/javascript">

var ewa;

function ewaStart(){

Ewa.EwaControl.add\_applicationReady(ewaConnect);

}

function ewaConnect() {

ewa = Ewa.EwaControl.getInstances().getItem(0);

if (ewa) {ewa.add\_activeCellChanged(ewaCellChanged);}

}

function ewaCellChanged(rangeArgs) {

var sheetName = rangeArgs.getRange().getSheet().getName();

var col = rangeArgs.getRange().getColumn();

var row = rangeArgs.getRange().getRow();

var value = rangeArgs.getFormattedValues();

alert('Active Cell is now at Row' + (row + 1) +

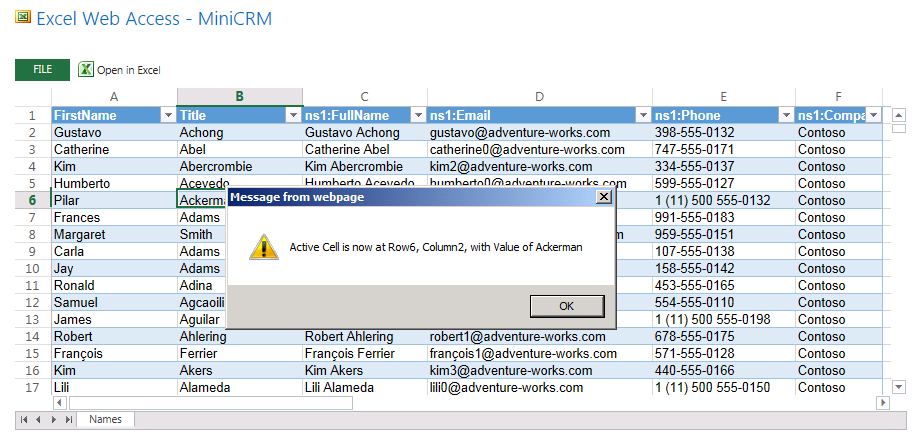
', Column' + (col + 1) + ', with Value of ' + value);

}

ewaStart();

</script>

1. In the ribbon, click **Stop Editing**.
2. Click on the spreadsheet and verify that a message is displayed showing the current active cell.



* + - * + Displaying Active Cell information

# Exercise 2: Excel Services REST

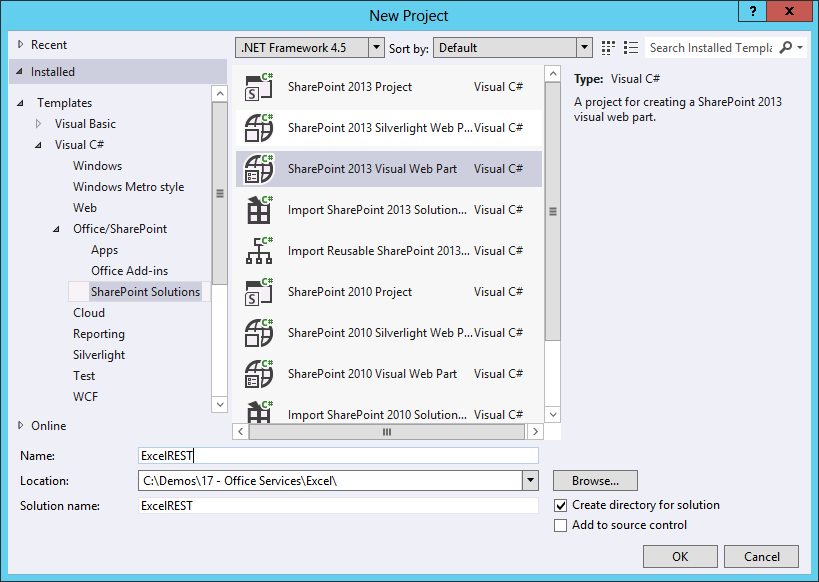
In this lab, you will create a web part that utilizes the RESTful interface to Excel Services.

## Task 1 – Create a Web Part

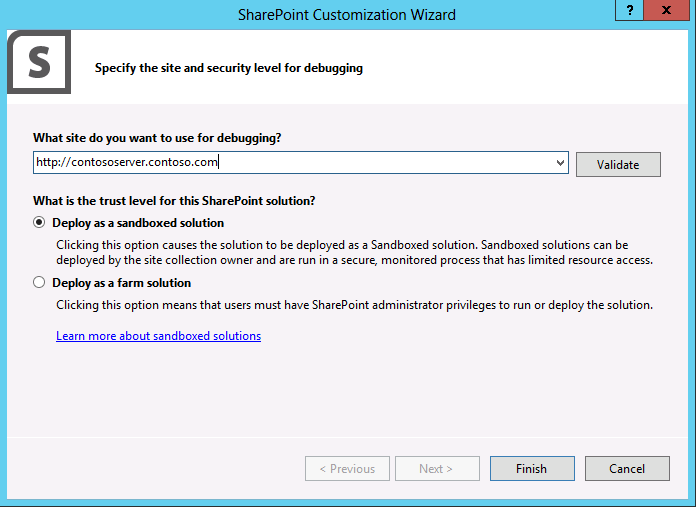
In this task, you will create a web part in a sandboxed solution.

* This task assumes the file MiniCRM.xlsx is located in the Shared Document library.
* You must also have the Sandboxed Code Service running.

1. Open Microsoft Visual Studio 2012 and create a new SharePoint project
   1. **Open** Microsoft Visual Studio 2012
   2. Select **File⮚New Project** from the main menu
   3. Click the **Templates⮚Visual C#⮚Office/SharePoint⮚SharePoint Solutions** node and select the **SharePoint 2013 Visual Web Part** project template
   4. Name the new project **ExcelREST**
   5. Click the **OK** button



* + - * + Create a New Visual Web Part project
  1. In the **SharePoint Customization Wizard**, enter the **URL** of the site to host the web part.
  2. Select **Deploy as Sandboxed Solution**.
  3. Click **Finish**.



* + - * + Deploy the project as a Sandboxed Solution

1. In the **Solution Explorer**, open **VisualWebPart1.ascx** for editing.
2. **Add** the following code to the file to make a RESTful call to the spreadsheet and display data.

<script type="text/javascript" language="javascript"

src="http://ajax.microsoft.com/ajax/jquery/jquery-1.7.1.min.js">

</script>

<script type="text/javascript">

$(document).ready(function () {

var e = ExecuteOrDelayUntilScriptLoaded(showSheet, "sp.js");

});

function showSheet() {

Results = {

element: '',

url: '',

init: function (element) {

Results.element = element;

Results.url = \_spPageContextInfo.webAbsoluteUrl +

"/\_vti\_bin/ExcelRest.aspx/Company%20Documents/MiniCRM.xlsx/OData/Table1";

},

load: function () {

$.ajax(

{

url: Results.url,

method: "GET",

headers: {

"accept": "application/json;odata=verbose",

},

success: Results.onSuccess,

error: Results.onError

}

);

},

onSuccess: function (data) {

var results = data.d.results;

var html = "<table>";

for (var i = 0; i < results.length; i++) {

html += "<tr><td>";

html += results[i].ns1FullName;

html += "</td><td>"

html += results[i].ns1Email;

html += "</td><tr>";

}

html += "</table>";

Results.element.html(html);

},

onError: function (err) {

alert(JSON.stringify(err));

}

}

Results.init($('#resultsDiv'));

Results.load();

}

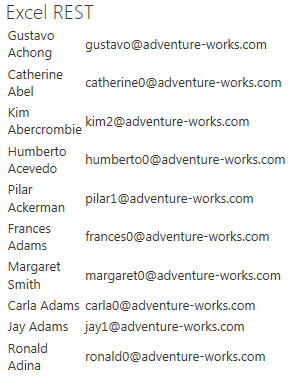
</script>

<div id="resultsDiv">Loading...</div>

## Task 2 – Deploy and Test the Web Part

In this task, you will deploy the web part in a sandboxed solution.

1. In the Solution Explorer, right click the **ExcelRest** project and select **Deploy**.
2. **Navigate** to the home page of the site where you deployed the web part.
3. Add the web part to the page.
   1. Click **Page** in the Ribbon.
   2. Click **Edit Page**.
   3. Click **Add a Web Part**.
   4. In the **Custom** folder, select **ExcelRest - VisualWebPart1**.
   5. Click **Add**.
   6. Click **Stop Editing**.
4. **Verify** that data appears in the page.



* + - * + Spreadsheet Data in a Web Part

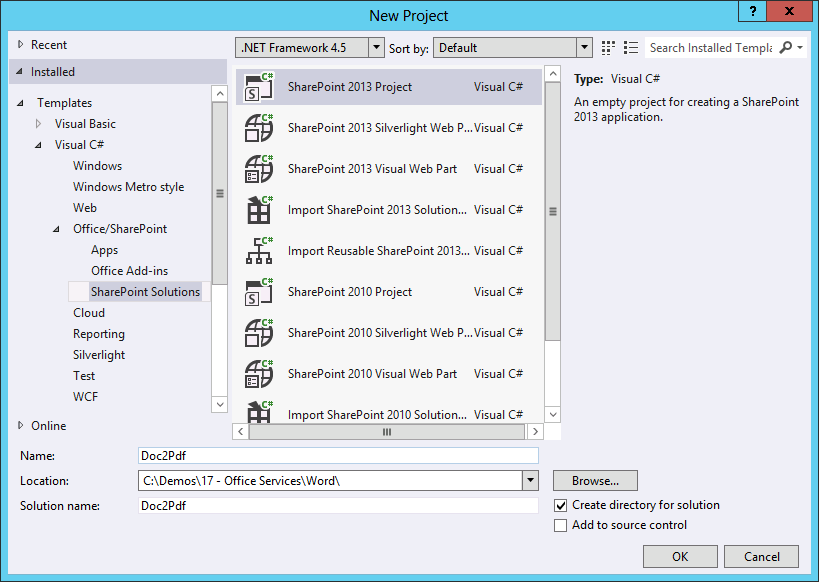
# Exercise 3: Word Automation Services

In this lab, you will create a feature for converting Word documents to PDF.

## Task 1 – Create the Converter Project

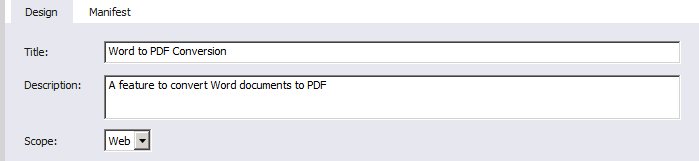
In this task, you will create the SharePoint project for converting Word documents to PDF.

1. Open Microsoft Visual Studio 2012 and create a new SharePoint project
   1. **Open** Microsoft Visual Studio 2012
   2. Select **File⮚New Project** from the main menu
   3. Click the **Templates⮚Visual C#⮚Office/SharePoint⮚SharePoint Solutions** node and select the **SharePoint 2013 Project** template
   4. Name the new project **Doc2Pdf**
   5. Click the **OK** button



* + - * + Create a new Empty SharePoint project
  1. In the **SharePoint Customization Wizard**, enter the **URL** of the site to host the web part.
  2. Select **Deploy as Farm Solution**.
  3. Click **Finish**.

1. Create a new Feature
   1. In the **Solution Explorer**, right click the **Features** node and select **Add Feature** from the context menu.
   2. Enter **Word to PDF Conversion** in the **Title**.
   3. Give the feature a description.
   4. Select **Web** as the **Scope**.



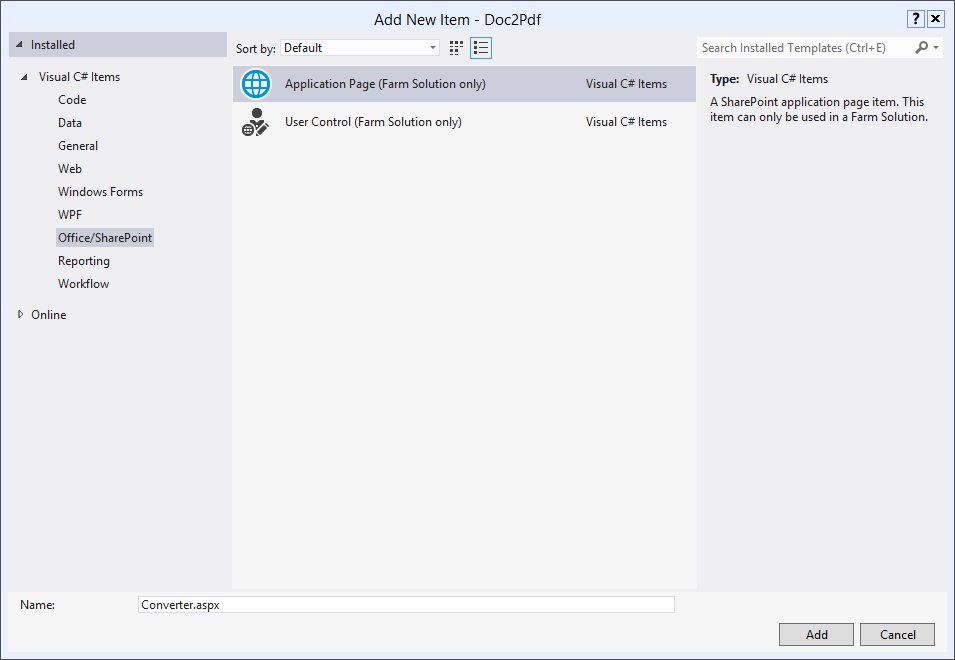
* + - * + Creating a new Feature

1. Add References
   1. In the **Solution Explorer**, right click the **References** node and select **Add Reference**.
   2. **Browse** to **\Program Files\Common Files\Microsoft Shared\web server extensions\15\ISAPI** and add a Reference to the following assemblies:

Microsoft.Office.Word.Server.dll

Microsoft.SharePoint.Security.dll

1. Add a LAYOUTS page to process the conversion.
   1. In the Solution Explorer, right click the **Doc2Pdf** node and select **Add⮚SharePoint Layouts Mapped Folder**.
   2. Right click the **Doc2Pdf** folder beneath the **Layouts** folder and select **Add⮚New Item** from the context menu.
   3. In the Add New Item dialog, select Application Page.
   4. Name the new page **Converter.aspx** and click **Add**.



* + - * + Add a new Application Page

1. Code the Application Page
   1. Open **Converter.aspx** for editing in Visual Studio.
   2. Add the following markup to **PlaceHolderMain**.

<div><asp:Literal ID="Messages" runat="server" />

* 1. Open **Converter.aspx.cs** for editing in Visual Studio.
  2. Add the following references to the top of the file.

using System.Text

using Microsoft.SharePoint.Security;

using Microsoft.Office.Word.Server.Conversions;

* 1. **Add** the following code within the **Page\_Load** method

try

{

string siteUrl = Request.QueryString["SiteUrl"];

string listId = Request.QueryString["ListId"];

string itemId = Request.QueryString["ItemId"];

SPSecurity.RunWithElevatedPrivileges(delegate()

{

using (SPSite site = new SPSite(siteUrl))

{

using (SPWeb web = site.OpenWeb())

{

SPDocumentLibrary library =

(SPDocumentLibrary)web.Lists[new Guid(listId)];

SPListItem item = library.GetItemById(int.Parse(itemId));

SPFile file = item.File;

if (file.Name.EndsWith(".doc",

StringComparison.CurrentCultureIgnoreCase) ||

file.Name.EndsWith(".docx",

StringComparison.CurrentCultureIgnoreCase))

{

//Set up the job

ConversionJobSettings jobSettings = new ConversionJobSettings();

jobSettings.OutputFormat = SaveFormat.PDF;

ConversionJob job =

new ConversionJob("Word Automation Services", jobSettings);

job.UserToken = web.CurrentUser.UserToken;

//File names

string wordFile = web.Url + "/" + item.Url;

string pdfFile = string.Empty;

if (file.Name.EndsWith(".doc",

StringComparison.CurrentCultureIgnoreCase))

pdfFile = wordFile.Replace(".doc", ".pdf");

if (file.Name.EndsWith(".docx",

StringComparison.CurrentCultureIgnoreCase))

pdfFile = wordFile.Replace(".docx", ".pdf");

//Start Job

job.AddFile(wordFile, pdfFile);

job.Start();

StringBuilder message = new StringBuilder();

message.Append("<p>The conversion job has been submitted</p>");

message.Append("<p>Word File: ");

message.Append(wordFile);

message.Append("</p><p>PDF File: ");

message.Append(pdfFile);

message.Append("</p>");

Messages.Text = message.ToString();

}

}

}

});

}

catch (Exception x)

{

Messages.Text = "<p>" + x.Message + "</p>";

}

1. Add the Custom Action.
   1. In the Solution Explorer, right click the **Doc2Pdf** project and select **Add⮚New Item** from the context menu.
   2. In the Add New Item dialog, select **Empty Element**.
   3. Name the new item **Action** and click **Add**.
   4. In the **Elements.xml** file, add the following **CAML**.

<CustomAction

Id="B8B3A0FD-CB68-4D3F-AB37-E5F4D83D1DBE"

RegistrationType="List"

RegistrationId="101"

Location="EditControlBlock"

Sequence="100"

Title="Convert Word to PDF">

<UrlAction

Url="{SiteUrl}/\_layouts/Doc2Pdf/Converter.aspx?

SiteUrl={SiteUrl}&amp;ListId={ListId}&amp;ItemId={ItemId}"/>

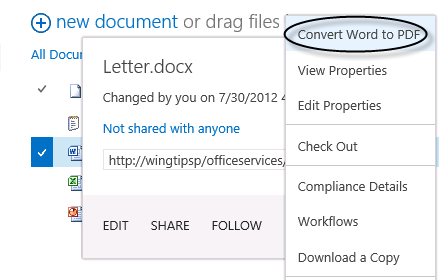
</CustomAction>

## Task 2 – Deploy and Test the Converter

In this task, you will deploy the new feature and test it on a Word document.

* You will need the file Letter.docx.

1. In the Solution Explorer, right click the **Doc2Pdf** project and select **Deploy** from the context menu.
2. Navigate to the **Company Documents** library you created earlier in the lab.
3. **Upload** the **Letter.docx** file.
4. Using the flyout menu associated with the file, select **More⮚Convert Word to PDF**.

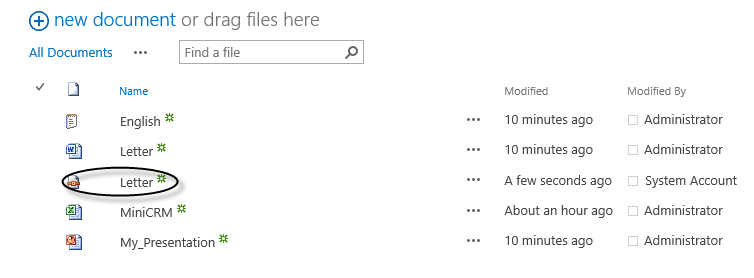


* + - * + Start the conversion process

1. The conversion process runs every 15 minutes by default. If you don’t want to wait, execute the following PowerShell command.

(Get-SPTimerJob “Word Automation Services”).Runnow()

1. **Verify** that a PDF version of the document appears in the library.



* + - * + Word document converted to PDF format

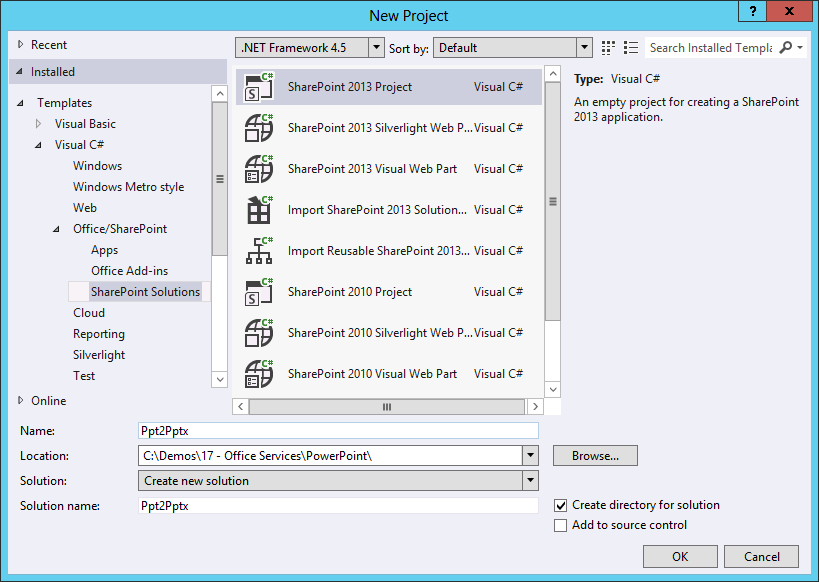
# Exercise 4: PowerPoint Automation Services

In this lab, you will create a feature for converting earlier versions of PowerPoint presentations in PPT format to the latest version in PPTX format.

## Task 1 – Create the Converter Project

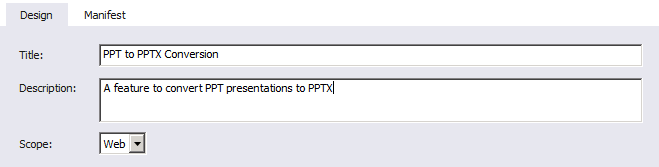
In this task, you will create the SharePoint project for converting PPT files to PPTX files.

1. Open Microsoft Visual Studio 2012 and create a new SharePoint project
   1. **Open** Microsoft Visual Studio 2012
   2. Select **File⮚New Project** from the main menu
   3. Click the **Templates⮚Visual C#⮚Office/SharePoint⮚SharePoint Solutions** node and select the **SharePoint 2013 Project** template
   4. Name the new project **Ppt2Pptx**
   5. Click the **OK** button



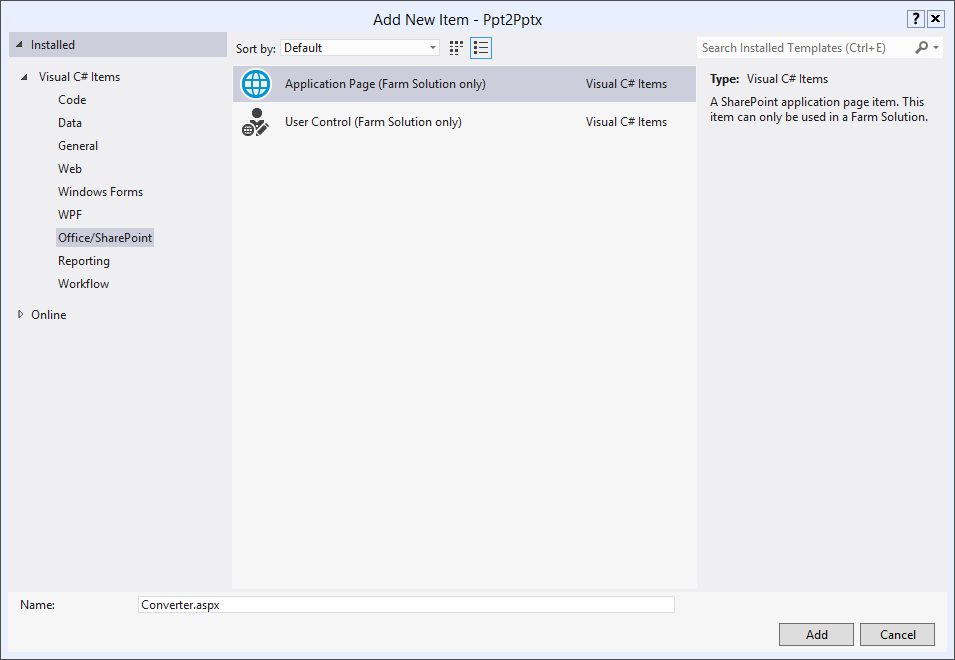
* + - * + Create a new Empty SharePoint project
  1. In the **SharePoint Customization Wizard**, enter the **URL** of the site to host the web part.
  2. Select **Deploy as Farm Solution**.
  3. Click **Finish**.

1. Create a new Feature
   1. In the **Solution Explorer**, right click the **Features** node and select **Add Feature** from the context menu.
   2. Enter **PPT to PPTX Conversion** in the **Title**.
   3. Give the feature a description.
   4. Select **Web** as the **Scope**.



* + - * + Creating a new Feature

1. Add References
   1. In the **Solution Explorer**, right click the **References** node and select **Add Reference**.
   2. **Browse** to **C:\Windows\Microsoft.NET\assembly\GAC\_MSIL\Microsoft.Office.Server.PowerPoint\v4.0\_15.0.0.0\_\_71e9bce111e9429c** and add a Reference to the assembly **Microsoft.Office.Server.PowerPoint.dll.**
2. Add a LAYOUTS page to process the conversion.
   1. In the Solution Explorer, right click the **Ppt2Pptx** node and select **Add⮚SharePoint Layouts Mapped Folder**.
   2. Right click the **Ppt2Pptx** folder beneath the **Layouts** folder and select **Add⮚New Item** from the context menu.
   3. In the Add New Item dialog, select Application Page.
   4. Name the new page **Converter.aspx** and click **Add**.



* + - * + Add a new Application Page

1. Code the Application Page
   1. Open **Converter.aspx** for editing in Visual Studio.
   2. Add the following markup to **PlaceHolderMain**.

<div><asp:Literal ID="Messages" runat="server" />

* 1. Open **Converter.aspx.cs** for editing in Visual Studio.
  2. Add the following references to the top of the file.

using System.Text

using System.IO;

using Microsoft.Office.Server.PowerPoint.Conversion;

using Microsoft.SharePoint.Administration;

* 1. **Add** the following member variables to the class:

string siteUrl = string.Empty;

string listId = string.Empty;

string itemId = string.Empty;

* 1. **Add** the following code within the **Page\_Load** method

string fileIn = string.Empty;

string fileOut = string.Empty;

try

{

siteUrl = Request.QueryString["SiteUrl"];

listId = Request.QueryString["ListId"];

itemId = Request.QueryString["ItemId"];

SPSecurity.RunWithElevatedPrivileges(delegate()

{

using (SPSite site = new SPSite(siteUrl))

{

using (SPWeb web = site.OpenWeb())

{

//Get item to convert

SPDocumentLibrary library = (SPDocumentLibrary)web.Lists[new Guid(listId)];

SPListItem item = library.GetItemById(int.Parse(itemId));

SPFile file = item.File;

//Get file names

fileIn = (string)file.Item[SPBuiltInFieldId.EncodedAbsUrl];

fileOut = fileIn.Replace(".ppt", ".pptx");

if (!fileIn.EndsWith(".ppt"))

throw new Exception("Input file must be PowerPoint 97-2003 file (.ppt)");

//Get file content to convert

byte[] buffer = file.OpenBinary();

MemoryStream inStream = new MemoryStream(buffer);

MemoryStream outStream = new MemoryStream();

// Create the presentation conversion request.

PresentationRequest request = new PresentationRequest(inStream, ".ppt", outStream);

// Synchronous Request

IAsyncResult result = request.BeginConvert(SPServiceContext.GetContext(site), null, null);

request.EndConvert(result);

// Add the converted file to the document library

web.AllowUnsafeUpdates = true;

web.Files.Add(fileOut, outStream.ToArray(), true);

web.AllowUnsafeUpdates = false;

//Go back to library

SPUtility.Redirect(library.DefaultViewUrl, SPRedirectFlags.Default, HttpContext.Current);

}

}

});

}

catch (Exception x)

{

StringBuilder message = new StringBuilder();

message.Append("<p>" + x.Message + "</p>");

message.Append("<p>PPT File: ");

message.Append(fileIn);

message.Append("</p><p>PPTX File: ");

message.Append(fileOut);

message.Append("</p>");

Messages.Text = message.ToString();

}

1. Add the Custom Action.
   1. In the Solution Explorer, right click the **Ppt2Pptx** project and select **Add⮚New Item** from the context menu.
   2. In the Add New Item dialog, select **Empty Element**.
   3. Name the new item **Action** and click **Add**.
   4. In the **Elements.xml** file, add the following **CAML**.

<CustomAction

Id="1D380367-4D45-4B48-834A-7A881DC5DC5D"

RegistrationType="List"

RegistrationId="101"

Location="EditControlBlock"

Sequence="100"

Title="Convert PPT to PPTX">

<UrlAction Url="{SiteUrl}/\_layouts/PPT2PPTX/Converter.aspx?

SiteUrl={SiteUrl}&amp;ListId={ListId}&amp;ItemId={ItemId}"/>

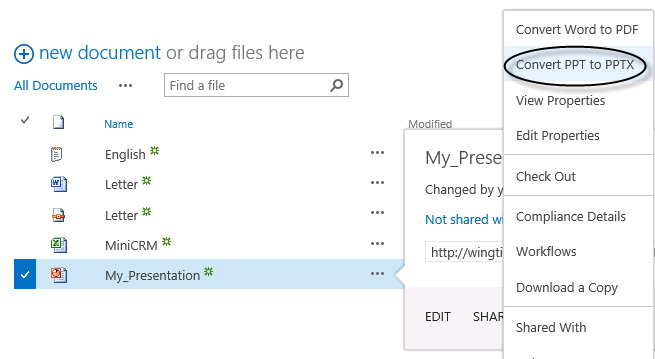
</CustomAction>

## Task 2 – Deploy and Test the Converter

In this task, you will deploy the new feature and test it on a PowerPoint document.

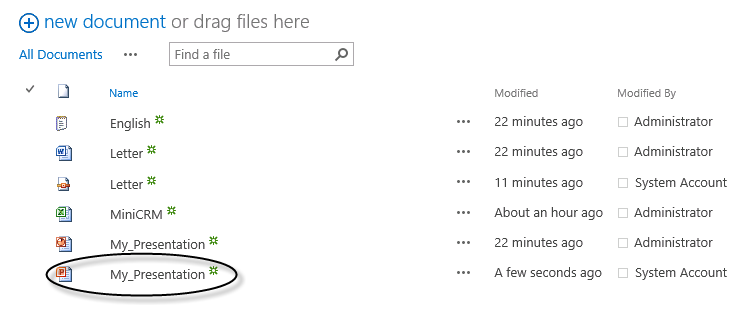
* You will need the file My\_Presentation.ppt.

1. In the Solution Explorer, right click the **Ppt2Pptx** project and select **Deploy** from the context menu.
2. Navigate to the **Company Documents** library you created earlier in the lab.
3. **Upload** the **My\_Presentation.ppt** file.
4. Using the flyout menu associated with the file, select **More⮚Convert PPT to PPTX**.



* + - * + Start the conversion process

1. **Verify** that a PPTX version of the document appears in the library.



* + - * + PPT document converted to PPTX format

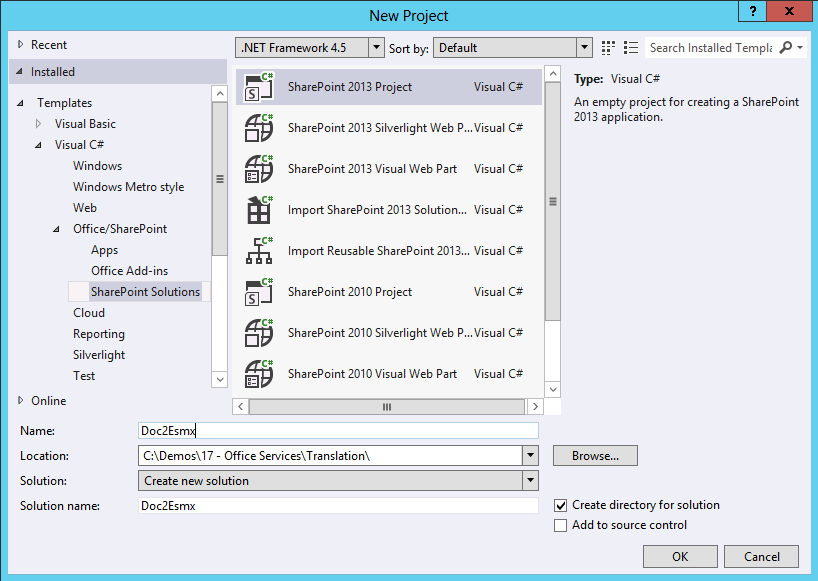
# Exercise 5: Translation Services

In this lab, you will create a feature for translating documents to Spanish.

## Task 1 – Create the Converter Project

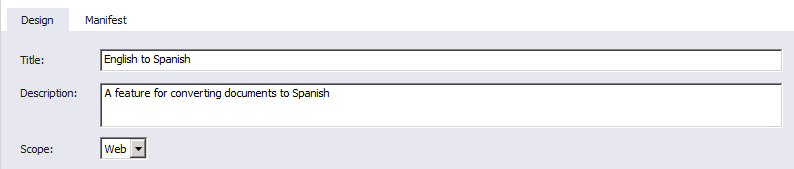
In this task, you will create the SharePoint project for converting documents to Spanish.

1. Open Microsoft Visual Studio 2012 and create a new SharePoint project
   1. **Open** Microsoft Visual Studio 2012
   2. Select **File⮚New Project** from the main menu
   3. Click the **Templates⮚Visual C#⮚Office/SharePoint⮚SharePoint Solutions** node and select the **SharePoint 2013 Project** template
   4. Name the new project **Doc2EsMx**
   5. Click the **OK** button



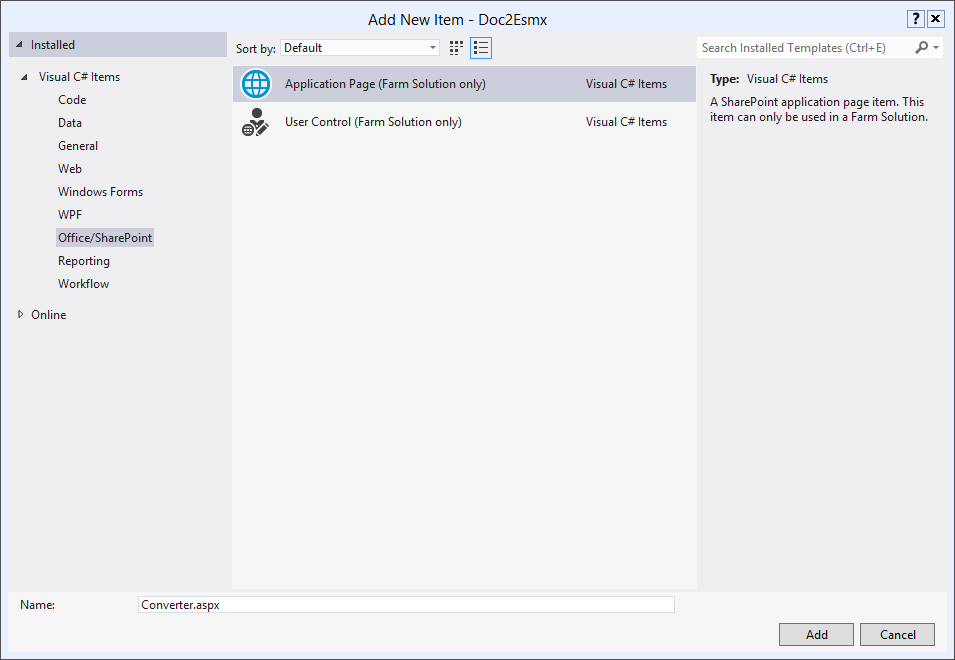
* + - * + Create a new Empty SharePoint project
  1. In the **SharePoint Customization Wizard**, enter the **URL** of the site to host the web part.
  2. Select **Deploy as Farm Solution**.
  3. Click **Finish**.

1. Create a new Feature
   1. In the **Solution Explorer**, right click the **Features** node and select **Add Feature** from the context menu.
   2. Enter **English to Spanish Conversion** in the **Title**.
   3. Give the feature a description.
   4. Select **Web** as the **Scope**.



* + - * + Creating a new Feature

1. Add References
   1. In the **Solution Explorer**, right click the **References** node and select **Add Reference**.
   2. **Browse** to **C:\Windows\Microsoft.NET\assembly\GAC\_MSIL\Microsoft.Office.TranslationServices\v4.0\_15.0.0.0\_\_71e9bce111e9429c** and add a Reference to the assembly **Microsoft.Office.TranslationServices.dll**
2. Add a LAYOUTS page to process the conversion.
   1. In the Solution Explorer, right click the **Doc2EsMx** node and select **Add⮚SharePoint Layouts Mapped Folder**.
   2. Right click the **Doc2EsMx** folder beneath the **Layouts** folder and select **Add⮚New Item** from the context menu.
   3. In the Add New Item dialog, select **Application Page**.
   4. Name the new page **Converter.aspx** and click **Add**.



* + - * + Add a new Application Page

1. Code the Application Page
   1. Open **Converter.aspx** for editing in Visual Studio.
   2. Add the following markup to **PlaceHolderMain**.

<div><asp:Literal ID="Messages" runat="server" />

* 1. Open **Converter.aspx.cs** for editing in Visual Studio.
  2. Add the following references to the top of the file.

using System.IO;

using System.Text;

using System.Web;

using System.Globalization;

using Microsoft.SharePoint.Utilities;

using Microsoft.Office.TranslationServices;

* 1. **Add** the following code within the **Page\_Load** method

string fileIn = string.Empty;

string fileOut = string.Empty;

try

{

string siteUrl = Request.QueryString["SiteUrl"];

string listId = Request.QueryString["ListId"];

string itemId = Request.QueryString["ItemId"];

SPSecurity.RunWithElevatedPrivileges(delegate()

{

using (SPSite site = new SPSite(siteUrl))

{

using (SPWeb web = site.OpenWeb())

{

//web.AllowUnsafeUpdates = true;

SPDocumentLibrary library = (SPDocumentLibrary)web.Lists[new Guid(listId)];

SPListItem item = library.GetItemById(int.Parse(itemId));

SPFile file = item.File;

//Get file names

fileIn = (string)file.Item[SPBuiltInFieldId.EncodedAbsUrl];

string extension = file.Name.Substring(file.Name.LastIndexOf('.') + 1);

fileOut = fileIn.Replace(extension, "\_esmx." + extension);

//Set up Job

SPServiceContext sc = SPServiceContext.GetContext(site);

byte[] inputByte = file.OpenBinary();

byte[] outputByte;

//Execute job synchronously

SyncTranslator job = new SyncTranslator(sc, CultureInfo.GetCultureInfo(2058));

//TranslationItemInfo itemInfo = job.Translate(fileIn, fileOut);

TranslationItemInfo itemInfo = job.Translate(inputByte, out outputByte, extension);

//Upload translated file

web.AllowUnsafeUpdates = true;

web.Files.Add(fileOut, outputByte, true);

web.AllowUnsafeUpdates = false;

//Go back to library

SPUtility.Redirect(library.DefaultViewUrl, SPRedirectFlags.Default, HttpContext.Current);

}

}

});

}

catch (Exception x)

{

StringBuilder message = new StringBuilder();

message.Append("<p>" + x.Message + "</p>");

message.Append("<p>Input File: ");

message.Append(fileIn);

message.Append("</p><p>Output File: ");

message.Append(fileOut);

message.Append("</p>");

Messages.Text = message.ToString();

}

1. Add the Custom Action.
   1. In the Solution Explorer, right click the **Doc2EsMx** project and select **Add⮚New Item** from the context menu.
   2. In the Add New Item dialog, select **Empty Element**.
   3. Name the new item **Action** and click **Add**.
   4. In the **Elements.xml** file, add the following **CAML**.

<CustomAction

Id="32FB3F24-AD65-4A25-A142-BB47A5127E02"

RegistrationType="List"

RegistrationId="101"

Location="EditControlBlock"

Sequence="100"

Title="Translate to Spanish">

<UrlAction Url="{SiteUrl}/\_layouts/Doc2EsMx/Converter.aspx?

SiteUrl={SiteUrl}&amp;ListId={ListId}&amp;ItemId={ItemId}"/>

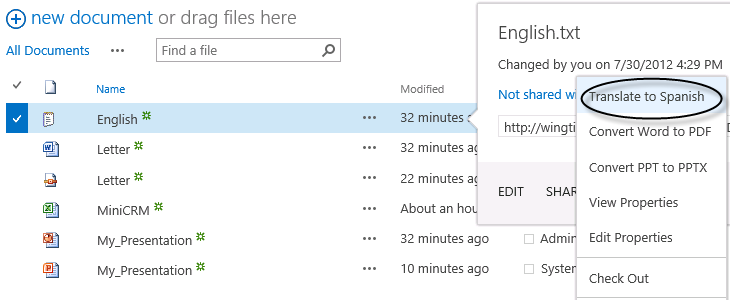
</CustomAction>

## Task 2 – Deploy and Test the Converter

In this task, you will deploy the new feature and test it on a text file.

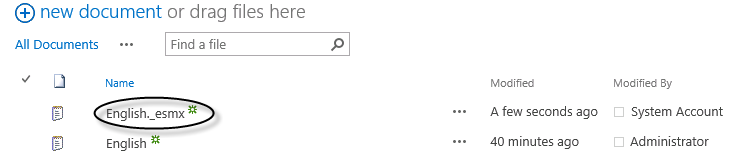
* You will need the file English.txt.

1. In the Solution Explorer, right click the **Doc2EsMx** project and select **Deploy** from the context menu.
2. Navigate to the **Shared Documents** library you created earlier in the lab.
3. **Upload** the **English.txt** file.
4. Using the flyout menu associated with the file, select **More⮚Translate to Spanish**.



* + - * + Start the translation process

1. **Verify** that a Spanish version of the document appears in the library.



* + - * + Text document translated to Spanish