

A SharePoint Developer Introduction

Hands-On Lab

Lab Manual

SPCHOL308 – Developing SharePoint 2010 User Interface Ribbon and Dialog Customizations – C#

This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2010 Microsoft. All rights reserved.

**Contents**

[SPCHOL308 – Developing SharePoint 2010 User Interface Ribbon and Dialog Customizations 2](#_Toc260142652)

[Lab Objective 2](#_Toc260142653)

[Additional Resources 2](#_Toc260142654)

[Getting Started 2](#_Toc260142655)

[Logging in to the Virtual Machine 2](#_Toc260142656)

[Locations 2](#_Toc260142657)

[Lab Pre-requisites 3](#_Toc260142658)

[Copying code samples from Word document 3](#_Toc260142659)

[Code Snippets 4](#_Toc260142660)

[Exercise 1 – Adding a custom button to the Ribbon 5](#_Toc260142661)

[Task 1 – View the site 5](#_Toc260142662)

[Task 2 – Create a SharePoint 2010 Empty Project 6](#_Toc260142663)

[Task 3 – Add the button code to the project 8](#_Toc260142664)

[Task 4 – Deploy the solution 13](#_Toc260142665)

[Exercise 2 – Accessing List Data using the JavaScript Client OM and displaying in a Dialog 16](#_Toc260142666)

[Task 1 – Create a new project and add a web part 16](#_Toc260142667)

[Task 2 - Add scripting code to access and render SharePoint list data. 19](#_Toc260142668)

[Task 3 - Deploy and test the web part 25](#_Toc260142669)

[Lab Summary 29](#_Toc260142670)

# SPCHOL308 – Developing SharePoint 2010 User Interface Ribbon and Dialog Customizations

Estimated time to complete this lab: **30 minutes**

Visual Studio 2010 and SharePoint Foundation 2010 are required for these exercises. These are installed on the Virtual Machine used in this lab.

## Lab Objective

The objective of this lab is to learn about how to modify the new SharePoint Ribbon menu and how to interact with the Client OM and the Dialog Framework.

* Extend the SharePoint Ribbon by adding a new button.
* Create a web part that utilizes client-side code using the new Client OM that will display a dialog box for users to enter data using the new Dialog Framework.

## Additional Resources

This lab includes the following additional resources:

|  |  |  |
| --- | --- | --- |
| This Lab Manual | SPCHOL308\_Manual\_CS.docx | This document |
| Source Code | Completed\CS\Ex1 Completed\CS\Ex2 | Completed lab source code in C#. |
| Resources | Resource\CS | Various resources used throughout this lab. |

## Getting Started

### Logging in to the Virtual Machine

Please log into the virtual machine as the following user:

**Username**: Administrator

**Password**: pass@word1

### Locations

This Hands-On Lab contains a number of additional resources in fixed locations. By default, it is assumed that the base HOL Resources directory is the **C:\Content Packs\Packs\SharePoint 2010 Developer Labs 1.0\Supporting Files\SPCHOL308** directory.

The default working folder for this lab is *“****\SPHOLS\SPCHOL308****”*.

### Lab Pre-requisites

Browse to base HOL directory ***Supporting Files\SPCHOL308\Resources***and execute the **optimize.ps1** PowerShell script:

1. Right-click on **optimize.ps1** and select **Run with PowerShell**:

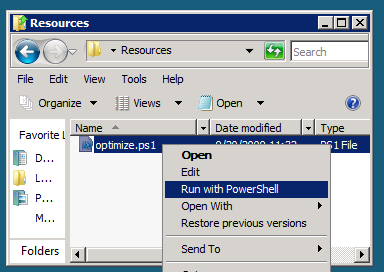


Figure 1 - Execute the PowerShell script

1. This will open a PowerShell window to execute the script. Please wait until the PowerShell script completes executing the script and closes the PowerShell window (this may take a few moments):

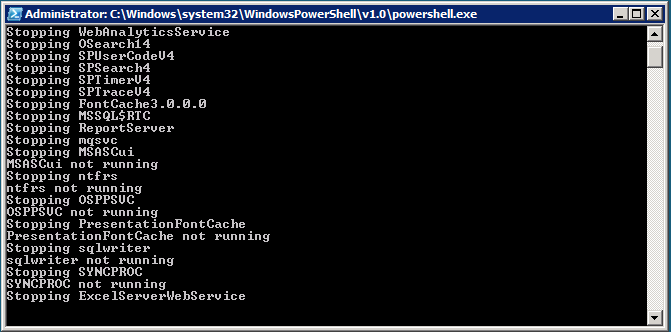


Figure 2 - PowerShell Window executing the script

### Copying code samples from Word document

Copying and pasting code from this Word document to Visual Studio is only possible for sections containing formatted code, e.g.:

Console.WriteLine("This is safe code!");

Code not in these sections may contain Unicode or other invisible characters that are not valid XML or C#/VB code, e.g.:

**Console.WriteLine(“This is NOT safe code !!”);**

### Code Snippets

You will also use Code Snippets to insert code during the lab.

To use the required code snippet:

* Right-click in the code file where you want to insert the code snippet.
* Select **Insert Snippet:**

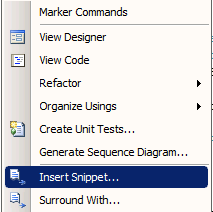
****

Figure 3 - Insert Snippet

Select a code snippet from the **My Code Snippets** gallery

## Exercise 1 – Adding a custom button to the Ribbon

In this exercise, we will extend the ribbon by adding our own custom button that executes server-side code. We can use this approach to add or replace buttons, groups, tabs, or even the whole ribbon. We will use a Visual Studio 2010 solution and deploy it to the local SharePoint server.

The button will pop up a JavaScript dialog box displaying the “Hello World” message.

### Task 1 – View the site

We will view the site before we add a button to the Ribbon so that we know what the Ribbon displays like OOTB.

1. Open Internet Explorer and browse to [**http://intranet.contoso.com**](http://intranet.contoso.com)
2. Select the **Shared Documents** library from the **Documents** menu in the left-hand navigation section (quick launch).

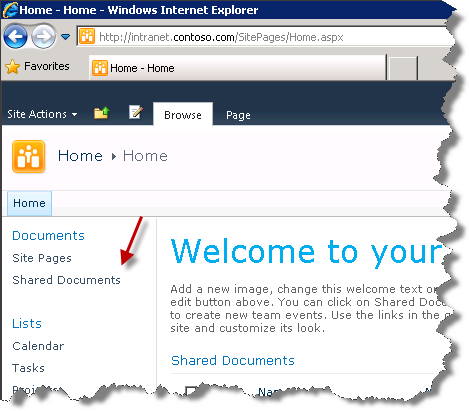


Figure 4 - Shared Documents

1. In the **Library Tools** tab of the Ribbon that appears, select the **Documents** tab.

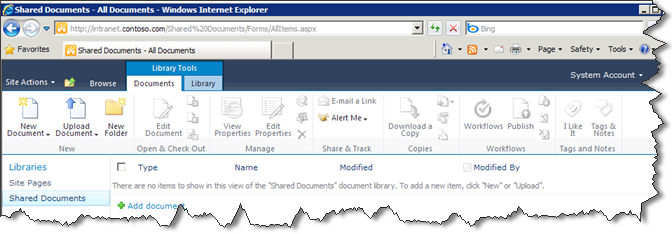


Figure 5 - Documents Tab

1. This Ribbon is the location where the new button we will be creating in this exercise will appear.

### Task 2 – Create a SharePoint 2010 Empty Project

In this task, a solution and project will be created. It will contain the rest of the development work in this exercise.

1. Open **Visual Studio 2010** by going to **the Start Menu | All Programs | Microsoft Visual Studio 2010 | Microsoft Visual Studio 2010**.
2. From the menu, select **File | New | Project**.
3. In the New Project dialog box, expand the **Installed Templates** left-hand menu to **Visual C# | SharePoint | 2010** and choose the **Empty SharePoint Project** project type in the project type list in the middle section of the screen.
4. Enter **RibbonDemo** in the Name textbox.
5. Enter **C:\SPHOLS\SPCHOL308\CS\Ex1**in the Location textbox.
6. Click **OK.**
7. A SharePoint Customization Wizard dialog box will appear.
8. In the What local site do you want to use for debugging? textbox, type [**http://intranet.contoso.com**](http://intranet.contoso.com)
9. For the What is the trust level for this SharePoint solution? radio buttons, choose **Deploy as a farm solution**.

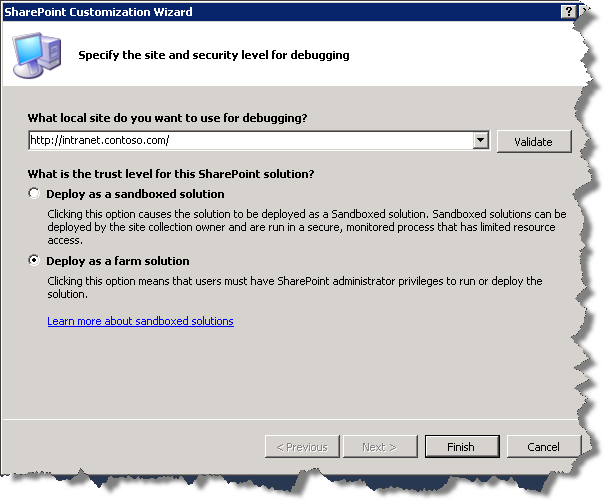


Figure 6 - SharePoint Customization Wizard

1. Press **Finish** to complete the customization wizard.
2. Visual Studio 2010 will generate the necessary project files and folders.
3. Your Solution Explorer should now look like the following image.

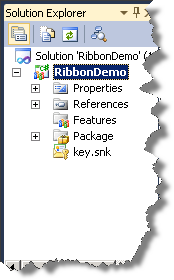


Figure 7 - RibbonDemo Solution Explorer

### Task 3 – Add the button code to the project

1. Right-click the **RibbonDemo** project node in Solution Explorer and choose **Add | New Item…**
2. In the Add New Item dialog box, expand the **Installed Templates** left-hand menu to **Visual C# | SharePoint | 2010** and choose the **Empty Element** item type in the middle section of the screen.
3. Leave the **Name** as *EmptyElement1*.

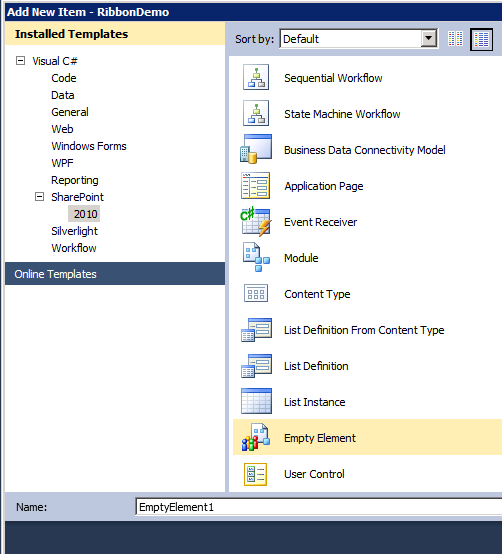


Figure 8 - Add New Item - Ribbon Demo

1. Click **Add** to add the element to your project.

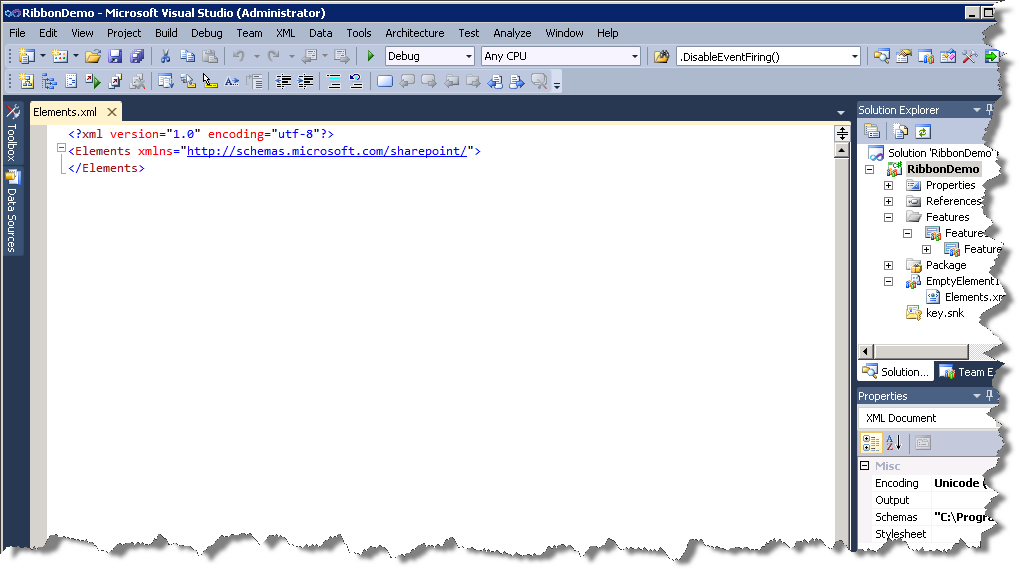


Figure 9 - Elements Xml

1. Delete the contents of the **Elements.xml** file.
2. Add the following code to the **Elements.xml** file. You can find a copy of this file in the **Supporting Files\SPCHOL308\Resources\CS** folder*.*

<?xml version="1.0" encoding="utf-8"?>

<Elements xmlns="http://schemas.microsoft.com/sharepoint/">

<CustomAction

Id="DemoHelloWorldButton"

RegistrationId="101"

RegistrationType="List"

Location="CommandUI.Ribbon"

Sequence="5"

Title="Hello World">

<CommandUIExtension>

<CommandUIDefinitions>

<CommandUIDefinition

Location="Ribbon.Documents.Manage.Controls.\_children">

<Button

Id="Ribbon.Documents.New.Controls.DemoHelloWorldButton"

Alt="Hello World Ribbon Button"

Sequence="10"

Command="Demo\_HelloWorld"

Image32by32="/\_layouts/images/ribbondemo/demobutton.png"

LabelText="Hello World Demo"

TemplateAlias="o1"/>

</CommandUIDefinition>

</CommandUIDefinitions>

<CommandUIHandlers>

<CommandUIHandler

Command="Demo\_HelloWorld"

CommandAction="javascript:alert('Hello World!');"

/>

</CommandUIHandlers>

</CommandUIExtension>

</CustomAction>

</Elements>

**Code Snippet**: *My Xml Snippets | spchol308\_ex1\_* *xml*



Figure 10 - Elements Xml Updated

1. Take note of the Image reference, this is the image that will display in the Ribbon, we will add this image next.
2. Take note also of the CommandScript, this is the JavaScript that will execute upon pressing of the button we are adding to the ribbon.
3. Add a folder to the solution that maps to the Images folder in the SharePoint layouts directory where we can put our button image. Right-click the **RibbonDemo** project node, and choose **Add | SharePoint “Images” Mapped Folder**.

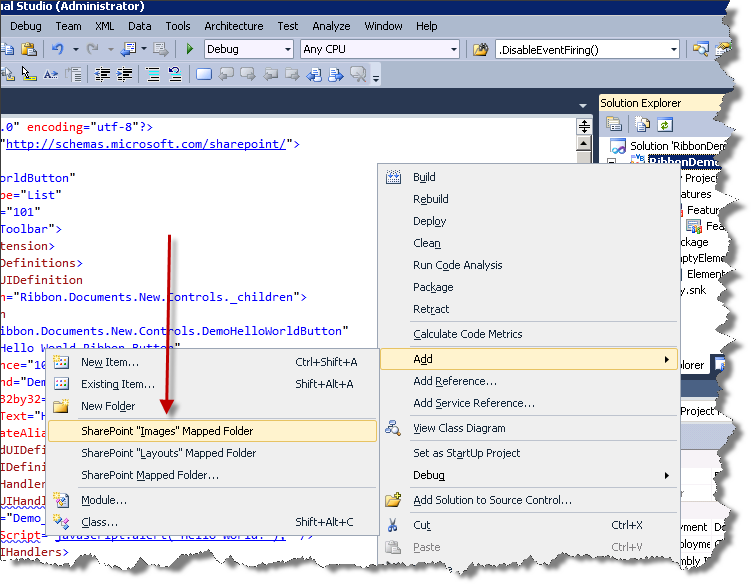


Figure 11 - Add Mapped Folder

1. The **Images** folder appears with a sub folder for our button.

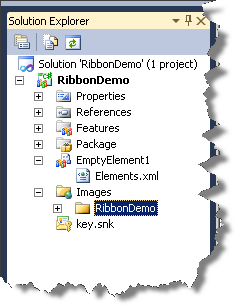


Figure 12 - Images Folder

1. Add the **DemoButton.png** file by right-clicking the new **RibbonDemo** folder and choosing **Add | Existing Item…**
2. When the dialog box appears navigate to **C:\Content Packs\Packs\SharePoint 2010 Developer Labs 1.0\SUPPORTING FILES\SPCHOL308\Resources\** and select the **DemoButton.png** file.

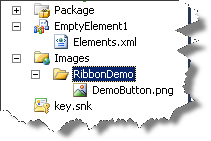


Figure 13 - DemoButton Image

### Task 4 – Deploy the solution

1. In the Solution Explorer, right-click on **RibbonDemo** and select **Deploy.**

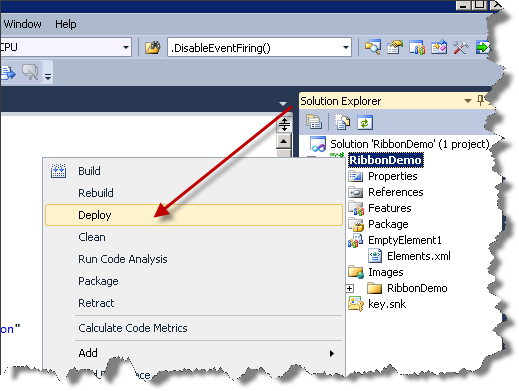


Figure 14 - Deploy Solution

1. The solution will be deployed to the SharePoint site.
2. Open a web browser and browse to the local SharePoint site:

[**http://intranet.contoso.com**](http://intranet.contoso.com)

1. If prompted for authentication, enter the following details:

**Username**: administrator

**Password**: pass@word1

1. In the left navigation, click on the **Shared Documents** link to open the *Shared Documents* library.

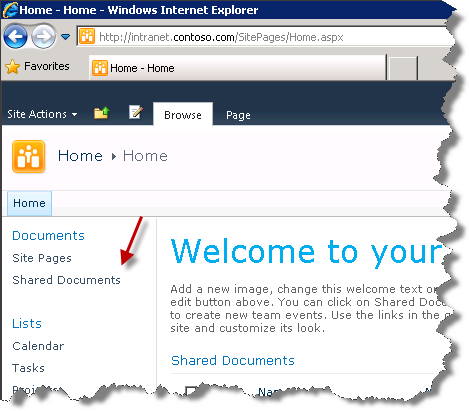


Figure 15 - Shared Documents link

1. Click on the **Documents** tabin the SharePoint Ribbon.

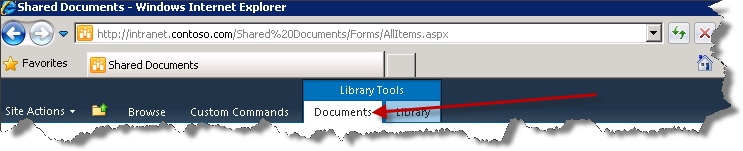


Figure 16 - Documents tab

1. You should see the new button **Hello World** added to the SharePoint Ribbon.

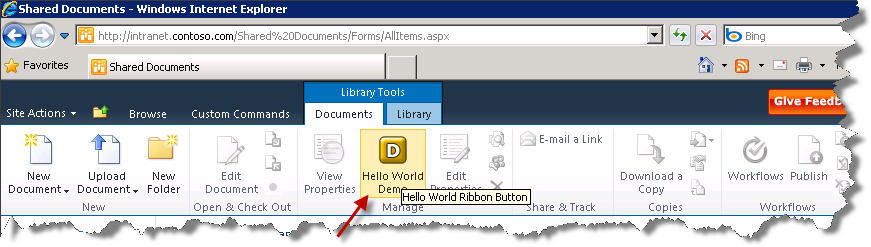


Figure 17 - Hello World Buton

1. Click the **Hello World Demo** button in the Ribbon to see the Hello World JavaScript dialog box appear.

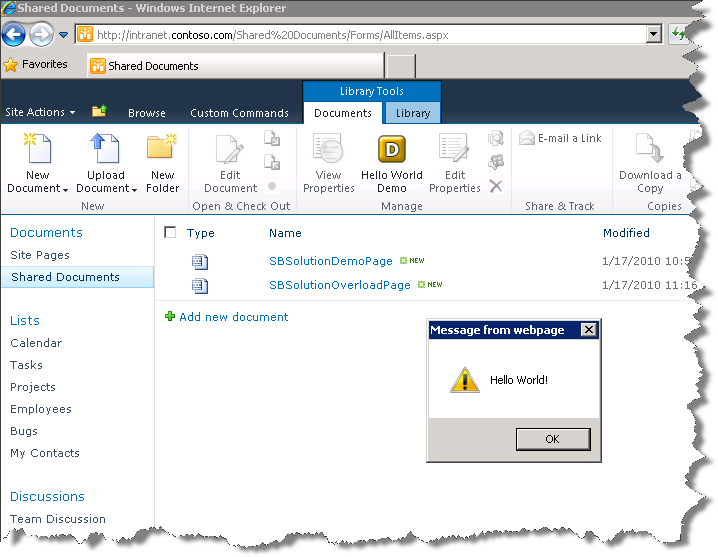


Figure 18 - Hello World Demo

In the past few minutes you have demonstrated how to add new custom button and extend the SharePoint button.

## Exercise 2 – Accessing List Data using the JavaScript Client OM and displaying in a Dialog

In SharePoint 2010, there are a number of object models that can be used by developers to access the server. The Client OM is a unified model which uses the same or similar concepts on the server, via web services and WCF services, via a client (JavaScript) API, and via REST. This paves the way for richer applications by dramatically simplifying accessing SharePoint data from client machines and other machines in the infrastructure.

In this exercise, we will use the EcmaScript client-side object model to update and read list data dynamically. Also shown is the new Dialog API’s from client-side script.

The JavaScript client side object model allows JavaScript developers access to SharePoint objects such as Site, Web and List (and more) from client side JavaScript.

### Task 1 – Create a new project and add a web part

In this task a solution and project will be created. It will contain the rest of the development work in this exercise.

1. Open **Visual Studio 2010** by going to **the Start Menu | Programs | Microsoft Visual Studio 2010 | Microsoft Visual Studio 2010**.
2. From the menu, select **File | New | Project**.
3. In the New Project dialog box, expand the **Installed Templates** left hand menu to **Visual C# | SharePoint | 2010** and choose the **Visual Web Part** project type in the project type list in the middle section of the screen.
4. Enter **ScriptOMandDialog** in the Name textbox.
5. Enter **C:\SPHOLS\SPCHOL308\CS\Ex2** in the Location textbox*.*

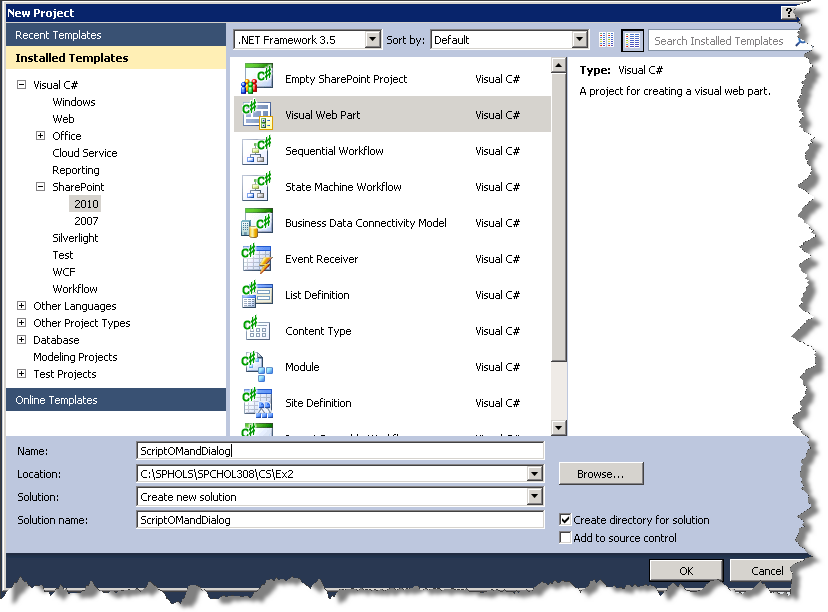


Figure 19 - New Project Dialog

1. Click **OK.**
2. A **SharePoint Customization Wizard dialog** box will appear.
3. In the **What local site do you want to use for debugging?** box, type [**http://intranet.contoso.com**](http://intranet.contoso.com)
4. From the **What is the trust level for this SharePoint solution?** radio buttons, choose **Deploy as a farm solution**.

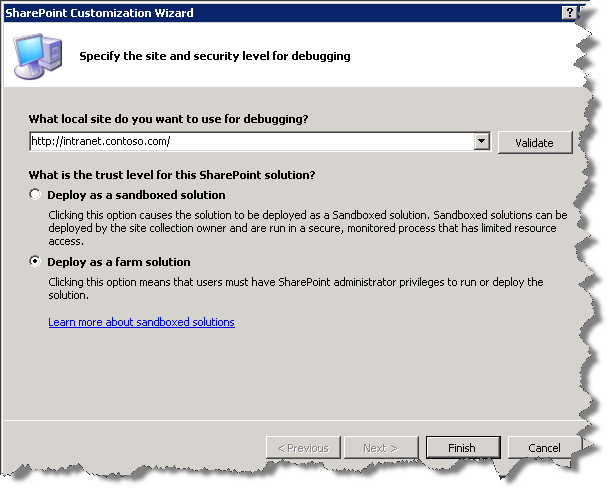


Figure 20 - SharePoint Customization Wizard

1. Press **Finish** to complete the customization wizard.
2. Visual Studio will create the **ScriptOMandDialog** solution and add the necessary files.

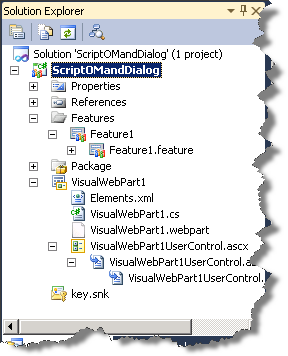


Figure 21 - Solution Explorer with Visual Web Part Project

1. In the Solution Explorer, double-click on the **VisualWebPart1.webpart** file.
2. Set the **Title** property to **ScriptOMandDialog**
3. Set the **Description** property to **ScriptOMandDialog description**

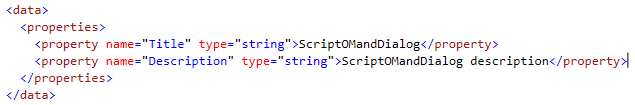


Figure 22 - Visual Web Part webpart file

1. Save and close the file.

### Task 2 - Add scripting code to access and render SharePoint list data.

1. In the Solution Explorer, **double-click** on **VisualWebPart1UserControl.ascx**
2. Add the following markup code at the end of the file.

<SharePoint:ScriptLink ID="ScriptLink1" runat="server" Name="sp.js" Localizable="false" LoadAfterUI="true" />

<script language="ecmascript" type="text/ecmascript">

</script>

**Code Snippet**: *My HTML Snippets | spchol308\_ex2\_*ScriptTags

1. Add the following JavaScript code within the Script block. You can either copy and paste all of the following code blocks or use **one** code snippet. This code declares important variables our code will use later on and adds our Initialize() function to be called after all of the SharePoint client side objects have loaded.

/\* SharePoint list names \*/

var ProjectListName = "Projects";

/\* SharePoint list field names \*/

var ProjectNameField = "Title";

var ProjectDescriptionField = "Description";

/\* List objects \*/

var projectsList;

/\* variable to hold list items from the projects list \*/

var projects;

/\* client context object - used to access SharePoint data \*/

var context;

/\* web (SPWeb) that our page is running on \*/

var web;

/\* variable to hold modalDialog to close later \*/

var modalDialog;

/\* used when creating a new project ListItem. \*/

var projectListItem;

var copyOfAddProjectForm;

/\* our startup method when the page loads \*/

\_spBodyOnLoadFunctionNames.push("Initialize()");

**Code Snippet**: *My JScript Snippets | spchol308\_ex2\_Jscript*

1. Our Initialize() function retrieves the SharePoint ClientContext object and loads up the projects list and initializes projects (Projects list items).

/\* Initialize useful variables and retrieve ClientContext \*/

function Initialize() {

/\* Retrieves the current ClientContext object \*/

context = SP.ClientContext.get\_current();

web = context.get\_web();

// Get references to the lists we will use

projectsList = web.get\_lists().getByTitle(ProjectListName);

var camlQuery = new SP.CamlQuery();

camlQuery.set\_viewXml('<View><Query/></View>');

projects = projectsList.getItems(camlQuery);

context.load(projects, 'Include(Title, Description)');

context.executeQueryAsync(onListsLoaded, OnError);

}

1. The onListsLoaded() event is called asynchronously from the Initialize() function.

/\* Event handler called loading the projects list

This method dynamically renders an HTML table to display the list data \*/

function onListsLoaded() {

// Get the list items for the contacts list

var projectTable = document.getElementById("tblProjectList");

// clear out the table

while (projectTable.rows.length > 0)

projectTable.deleteRow(projectTable.rows.length - 1);

var content;

var cell;

var tbo = document.createElement('tbody');

// Loop for each project

var listItemEnumerator = projects.getEnumerator();

while (listItemEnumerator.moveNext()) {

// For each project create a row in the table

var newTR = document.createElement('tr');

var projectLI = listItemEnumerator.get\_current();

// get\_item() retrieves the listitem value

var projectName =

projectLI.get\_item(ProjectNameField);

var projectDesc =

projectLI.get\_item(ProjectDescriptionField);

// add the cells for the row

cell = document.createElement('td');

content = document.createTextNode(projectName);

cell.appendChild(content);

newTR.appendChild(cell);

cell = document.createElement('td');

content = document.createTextNode(projectDesc);

cell.appendChild(content);

newTR.appendChild(cell);

// Add the row to the table body

tbo.appendChild(newTR);

}

// add the table body to the table

projectTable.appendChild(tbo);

}

1. The ShowAddProject() function finds the divAddProjectElement (which we’ll add later) and displays it using the ModalDialog.showModalDialog() method.

/\* Hide the modal dialog and display the updated UI \*/

function onProjectAdded() {

HideAddProject();

}

/\* Show a modalDialog with the contents of divAddProject \*/

function ShowAddProject() {

var divAddProject = document.getElementById("divAddProject");

// showModalDialog removes the element passed in from the DOM

// so we save a copy and add it back later

copyOfAddProjectForm = divAddProject.cloneNode(true);

divAddProject.style.display = "block";

var options = { html: divAddProject, width: 200, height: 350, dialogReturnValueCallback: ReAddClonedForm };

modalDialog = SP.UI.ModalDialog.showModalDialog(options);

}

/\* Close the modalDialog \*/

function HideAddProject() {

modalDialog.close();

Initialize();

}

function ReAddClonedForm() {

document.body.appendChild(copyOfAddProjectForm);

}

1. The AddProject() function creates a new list item in the Project List*.*

/\* Called from the Add Project modal dialog

Creates a list item in the Project list \*/

function AddProject() {

var lici1 = new SP.ListItemCreationInformation();

projectListItem = projectsList.addItem(lici1);

projectListItem.set\_item(ProjectNameField, getTBValue("txtProjectName"));

projectListItem.set\_item(ProjectDescriptionField, getTBValue("txtDescription"));

projectListItem.update();

context.load(projectListItem);

// Execute the query to create the project list

// onProjectAdded is our call back method called when the call to the server has completed

context.executeQueryAysnc(onProjectAdded, OnError);

}

1. Helper functions / Error handler

/\* Error handler \*/

function OnError(sender, args) {

var spnError = document.getElementById("spnError");

spnError.innerHTML = args.get\_message();

}

/\* Helper function - shortcut to the value property of a textbox \*/

function getTBValue(elID) {

var el = document.getElementById(elID);

return el.value;

}

1. Add the following HTML code below the ending script block tag (</script>). *tblProjectList* is an empty table that our script dynamically adds rows/columns to for each project. There is also a link to invoke ShowAddProject(), which will launch a modal dialog box. The markup within *divAddProject* will be displayed in a modal dialog that allows the user to input data to create a new project*.*

<div style="font-weight: bold">Project List</div>

<br />

<table id="tblProjectList" style="border: solid 1px silver">

</table>

<br />

<a href="javascript:ShowAddProject()">Add a project</a>

<br />

<div id="divAddProject" style="display: none; padding: 5px">

<b>Project Information</b><br /><br />

Name <br />

<input type="text" id="txtProjectName" /><br />

Description<br />

<input type="text" id="txtDescription" /><br />

<span id="spnError" style="color: Red" /><br />

<input type="button" value="Add New Project" onclick="AddProject()" />

</div>

**Code Snippet**: My HTML Snippets | spchol308\_ex2\_HTML

### Task 3 - Deploy and test the web part

1. Right-click on the **ScriptOMandDialog** project node and select **Deploy**.

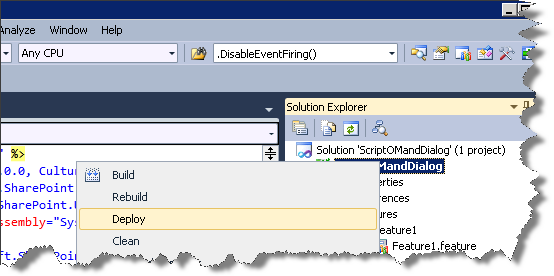


Figure 23 - Deploy

1. When deployment is complete (you can watch the progress in the Output window and see the status updated in the status bar), add the web part to a web part page:
2. Open Internet Explorer and browse to [**http://intranet.contoso.com**](http://intranet.contoso.com)
3. Click the **Edit** icon to view the Edit ribbon.

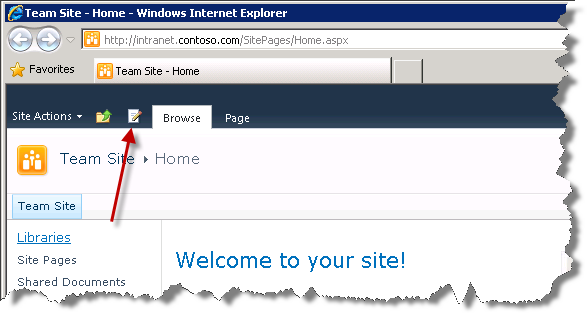


Figure 24 – Edit Icon

1. Select **Insert** from the **Editing Tools** tab on the Edit ribbon

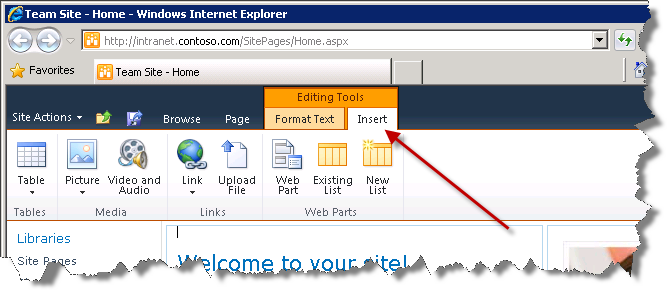


Figure 25 - Format Text Tab

1. Click the **Web Part** button on the **Insert** ribbon
2. Choose **Custom** from the **Categories** column
3. Choose **ScriptOMandDialog** from the **Web Parts** column

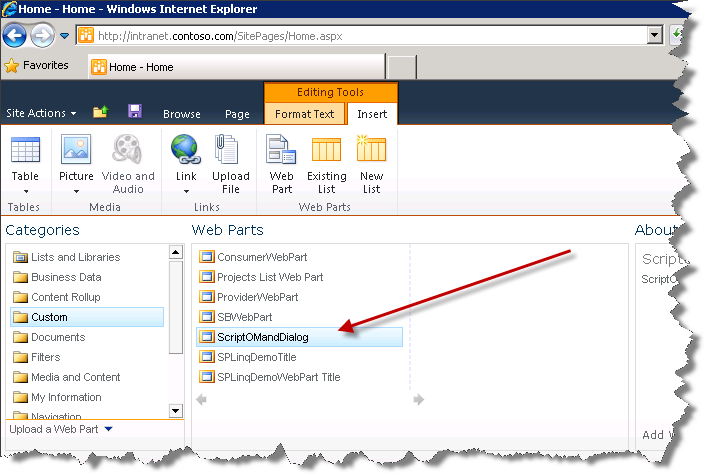


Figure 26 - Insert Web Part

1. Click the **Add** button at the bottom right of the Add Web Part area

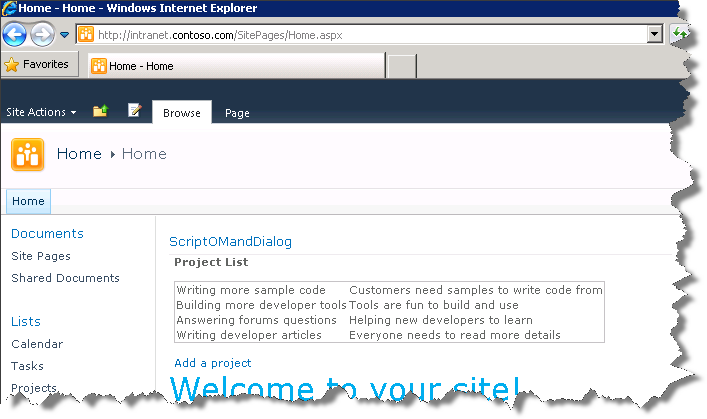


Figure 27 - ScriptOMandDialog Web Part

1. Click **Page** in the ribbon.
2. Click the down arrow under the **Save & Close** button on the ribbon, then select **Stop Editing**.
3. Click **Yes** when you see the Save Changes dialog. Your ScriptOMandDialog web part should now be displayed on the page.
4. Click the **Add a project** link (directly under the new web part), fill in the Name and Description fields as you wish, then click the **Add New Project** button.

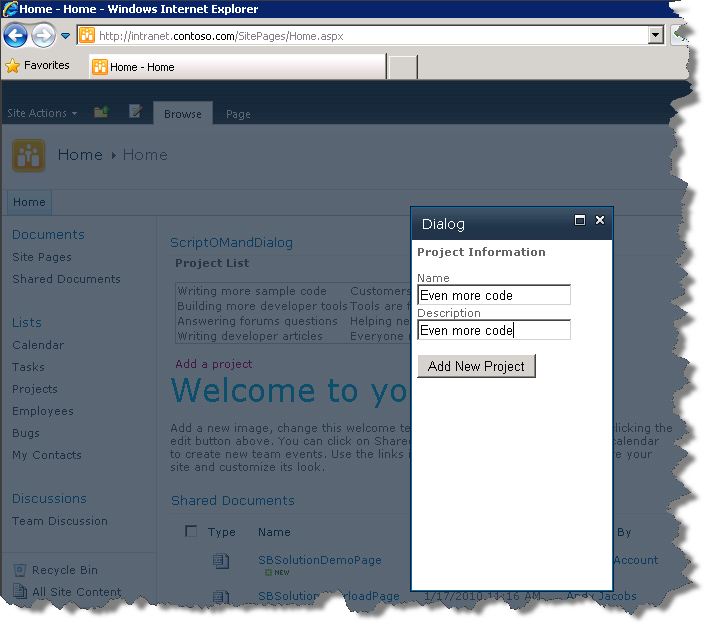


Figure Sample 28 - Add a project

1. You should see the new project with the details you specified appear in the **Project List**.

In the past few minutes, you have demonstrated how to use the JavaScript Client Object Model and build & deploy a Visual Web Part.

## Lab Summary

In this lab you performed the following exercises.

Created new SharePoint 2010 project of type Empty

Added an Element.xml file to a SharePoint project containing a Custom Action to extend the Ribbon menu

Added a SharePoint *Images* mapped folder which maps to the Images folder in the SharePoint 2010 directory.

Deployed the Empty project type containing one feature to your SharePoint site

Created new SharePoint 2010 project type of a Visual Web Part

Added client-side script to the web part to render list data

Added client-side script and HTML to the web part to utilize the Dialog Framework

Deployed a SharePoint Visual Web Part solution