

Programmatically Add an Excel Web Access Web Part to a Page in SharePoint 2010

(Microsoft Corporation)

This example shows how to programmatically add an Excel Web Access Web Part to a SharePoint page. It also shows you how to display an Excel workbook programmatically in an Excel Web Access Web Part. The following project uses Microsoft Visual Studio.

Note: Depending on the Visual Studio version and the Visual Studio integrated development environment (IDE) settings that you are using, the process and steps to create a Visual Studio project could be slightly different from the procedures shown in this topic.

Note: It is assumed that you have already created a SharePoint document library and made it a trusted location.

Adding a Reference

The following steps show how to locate Microsoft.Office.Excel.WebUI.dll and how to add a reference to it. Repeat for Microsoft.Office.Excel.WebUI.Internal.dll and Microsoft.SharePoint.dll.

Note: It is assumed that you have already copied Microsoft.Office.Excel.WebUI.dll and Microsoft.Office.Excel.WebUI.Internal.dll from the global assembly cache to a folder of your choice. For more information about how to locate and copy Microsoft.Office.Excel.WebUI.dll and Microsoft.Office.Excel.WebUI.Internal.dll, see How to: Locate and Copy Microsoft.Office.Excel.WebUI.dll and Microsoft.Office.Excel.WebUI.Internal.dll2.

To add a reference to Microsoft.Office.Excel.WebUI.dll

1. On the Project menu, click Add Reference.
2. In the Add Reference dialog box, click Browse.
3. Browse to the location of Microsoft.Office.Excel.WebUI.dll.
4. Select Microsoft.Office.Excel.WebUI.dll, and then click OK.
5. Click Add Reference. A reference to Microsoft.Office.Excel.WebUI.dll is added to your project.

Note: You can also open the Add Reference dialog box in the Solution Explorer pane by right-clicking References and selecting Add Reference.

Instantiating a Web Part

To instantiate the Excel Web Access Web Part

1. Add the Microsoft.Office.Excel.WebUI namespace as a directive to your code, so that when you use the types in this namespace, you do not need to fully qualify them:

```
using Microsoft.Office.Excel.WebUI;
```

2. Instantiate and initialize the Excel Web Access Web Part, as follows:

```
ExcelWebRenderer ewaWebPart = new ExcelWebRenderer();
```

To display a workbook programmatically

Programmatically Add an Excel Web Access Web Part to a Page in ShaePoint 2010

(Microsoft Corporation)

In this example, the AddWebPart method takes in the path to an Excel workbook location as an argument. The user provides the path by typing in a Windows Forms text box and clicking a button.

```
publicbool AddWebPart(string sitename, string book)
{
    ...
}

privatevoid AddEWAButton_Click(object sender,
    EventArgs e)
{
    siteurl = textBox1.Text;
    bookuri = textBox2.Text;
    succeeded = AddWebPart(siteurl, bookuri);
    if (succeeded)
    {
        MessageBox.Show(
            success,
            appname,
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
        progressBar1.Value = 1;
    }
}
```

Important: Ensure that the location where the workbook is saved is a trusted location.

You can display an Excel workbook programmatically by using the following code.

```
// Instantiate Excel Web Access Web Part.// Add an Excel Web Access Web Part in a
shared view.
ExcelWebRenderer ewaWebPart = new ExcelWebRenderer();
ewaWebPart.WorkbookUri = book;
progressBar1.PerformStep();

try
{
    webPartManager.AddWebPart(ewaWebPart, "Left", 0);
}
catch (Exception exc)
{
    MessageBox.Show(
        addWebPartError + "\n" + exc.Message,
        appName,
        MessageBoxButtons.OK,
        MessageBoxIcon.Asterisk);
    progressBar1.Value = 1;
    return b;
}
```

Programmatically Add an Excel Web Access Web Part to a Page in ShaePoint 2010

(Microsoft Corporation)

Example

The following example is a Windows Forms application that enables a user to enter information on a SharePoint site and display an Excel workbook saved in a trusted location programmatically. It programmatically creates an Excel Web Access Web Part on the default.aspx page of the specified site and displays the specified Excel workbook.

The code sample is the code from the Form1.cs and Form1.vb example files described in the previous procedures. The code sample uses two text boxes, a progress bar, and a button. The code is only a portion of the Windows Forms project. For example, the code involving the layout of the form is not shown.

```
namespace AddEWATool
{
    using System;
    using System.Windows.Forms;
    using Microsoft.Office.Excel.WebUI;
    using Microsoft.SharePoint;
    using Microsoft.SharePoint.WebPartPages;

    ///<summary>/// Form1 class derived from
    System.Windows.Forms.///</summary>publicpartialclass Form1 : Form
    {
        privatestring appName = "AddEWATool";
        privatestring specifyInputError = "Please add a site URL, for example:
http://myserver/site/";
        privatestring openSiteError = "There was a problem with the site name. Please
check that the site exists.";
        privatestring addWebPartError = "There was a problem adding the Web Part.";
        privatestring successMessage = "Web Part successfully added.";

        ///<summary>/// Add the Excel Web Access Web Part to the Default.aspx page of
the specified site.///</summary>///<param name="siteName">URL of the SharePoint
site</param>///<param name="book">URI to the workbook</param>///<returns>Returns true
if the WebPart was successfully added; otherwise, false.</returns>publicbool
AddWebPart(string siteName, string book)
    {
        SPSite site = null;
        SPWeb targetWeb = null;
        SPLimitedWebPartManager webPartManager = null;

        bool b = false;
        progressBar1.Visible = true;
        progressBar1.Minimum = 1;
        progressBar1.Maximum = 4;
        progressBar1.Value = 1;
        progressBar1.Step = 1;
    }
}
```

Programmatically Add an Excel Web Access Web Part to a Page in ShaePoint 2010

(Microsoft Corporation)

```
if (String.IsNullOrEmpty(siteName))
{
    MessageBox.Show(
        specifyInputError,
        appName,
        MessageBoxButtons.OK,
        MessageBoxIcon.Asterisk);
    return b;
}

try
{
    try
    {
        site = new SPSSite(siteName);
        targetWeb = site.OpenWeb();
    }
    catch (Exception exc)
    {
        MessageBox.Show(
            openSiteError + "\n" + exc.Message,
            appName,
            MessageBoxButtons.OK,
            MessageBoxIcon.Asterisk);
        progressBar1.Value = 1;
        return b;
    }

    progressBar1.PerformStep();

    try
    {
        // Get the shared Web Part manager on the Default.aspx page.
        webPartManager = targetWeb.GetLimitedWebPartManager(
            "Default.aspx",

System.Web.UI.WebControls.WebParts.PersonalizationScope.Shared);
    }
    catch (Exception exc)
    {
        MessageBox.Show(
            openSiteError + "\n" + exc.Message,
            appName,
            MessageBoxButtons.OK,
            MessageBoxIcon.Asterisk);
        progressBar1.Value = 1;
        return b;
    }
}
```

Programmatically Add an Excel Web Access Web Part to a Page in ShaePoint 2010

(Microsoft Corporation)

```
    }

    progressBar1.PerformStep();

    // Instantiate Excel Web Access Web Part.// Add an Excel Web Access
Web Part in a shared view.
    ExcelWebRenderer ewaWebPart = new ExcelWebRenderer();
    ewaWebPart.WorkbookUri = book;
    progressBar1.PerformStep();

    try
    {
        webPartManager.AddWebPart(ewaWebPart, "Left", 0);
    }
    catch (Exception exc)
    {
        MessageBox.Show(
            addWebPartError + "\n" + exc.Message,
            appName,
            MessageBoxButtons.OK,
            MessageBoxIcon.Asterisk);
        progressBar1.Value = 1;
        return b;
    }
}
finally
{
    if (site != null)
    {
        site.Dispose();
    }

    if (targetWeb != null)
    {
        targetWeb.Dispose();
    }

    if (webPartManager != null)
    {
        webPartManager.Dispose();
    }
}

    progressBar1.PerformStep();
    b = true;
    return b;
}
```

Programmatically Add an Excel Web Access Web Part to a Page in ShaePoint 2010

(Microsoft Corporation)

```
///<summary>/// AddEWAButton click handler.///</summary>///<param
name="sender">caller</param>///<param name="e">event</param>privatevoid
AddEWAButton_Click(object sender, EventArgs e)
{
    string siteUrl = textBox1.Text;
    string bookUri = textBox2.Text;
    bool succeeded = AddWebPart(siteUrl, bookUri);
    if (succeeded)
    {
        MessageBox.Show(
            successMessage,
            appName,
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
        progressBar1.Value = 1;
    }
}
}
```