Using Excel Web Services in a SharePoint Web Part

Dan Battagin, Microsoft Corporation

November 2006

**Applies to:** Microsoft Office SharePoint Server 2007, Microsoft Office Excel 2007

**Summary:** Learn how to create a Web Part that can be deployed to Microsoft Office SharePoint Server 2007 and call Excel Web Services to perform custom calculations. (13 printed pages)

Download: [MOSS2007ExcelWebPart.exe](http://www.microsoft.com/downloads/details.aspx?familyid=2c5919e4-0757-4db1-9798-e0e4032f12a8&displaylang=en)

Watch [SharePoint 2007 Video: Excel Services Mortgage Calculator](http://wm.microsoft.com/ms/msdn/office/sharepoint2007excelservicesmortgagecalc/sharepoint2007excelservicesmortgagecalc.wmv).

**Contents**

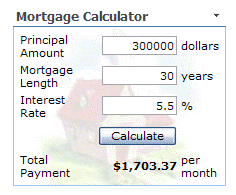
* [Overview of Using Excel Web Services in a SharePoint Web Part](http://msdn2.microsoft.com/en-us/library/aa973804.aspx#Office2007ExcelServicesWebPart_Intro)
* [Creating a Web Control Project](http://msdn2.microsoft.com/en-us/library/aa973804.aspx#Office2007ExcelServicesWebPart_WebControlProject)
* [Creating the Mortgage Calculator Form](http://msdn2.microsoft.com/en-us/library/aa973804.aspx#Office2007ExcelServicesWebPart_Form)
* [Calling the Excel Web Service to Perform the Mortgage Calculation](http://msdn2.microsoft.com/en-us/library/aa973804.aspx#Office2007ExcelServicesWebPart_ExcelWebServices)
* [Conclusion](http://msdn2.microsoft.com/en-us/library/aa973804.aspx#Office2007ExcelServicesWebPart_Conclusion)
* [Additional Resources](http://msdn2.microsoft.com/en-us/library/aa973804.aspx#Office2007ExcelServicesWebPart_AdditionalResources)

**Overview of Using Excel Web Services in a SharePoint Web Part**

ASP.NET Web Part controls are an integrated set of controls for creating Web sites that enable end users to modify the content, appearance, and behavior of Web pages directly in a browser.

This article walks you through a series of steps that show how to create a mortgage calculator Web Part, as shown in Figure 1.

**Figure 1. Mortgage calculator Web Part that calls Excel Services to perform calculation**



You use Microsoft Visual Studio 2005 to create a new Web control project that contains the code and supporting files for your Web Part.

After you add the necessary code to that project to display the Web Part and call Excel Web Services, the next step is to build a deployment project. This project is used to deploy your mortgage calculator Web Part to a Microsoft Office SharePoint Server 2007 site by using the administrative tools that are provided with Microsoft Windows SharePoint Services.

To learn how to package and deploy the Web Part that you create in this walkthrough, and how to improve the appearance and configurability of a Web Part, see [Deploying and Optimizing a SharePoint Web Part That Calls Excel Web Services](http://msdn2.microsoft.com/en-us/library/aa973805.aspx).

**System Requirements**

To create and run the samples, you must have the following software installed on your computer:

* Microsoft Visual Studio 2005
* Microsoft Office Excel 2007
* Office SharePoint Server 2007

**Creating a Web Control Project**

To begin, you must first create a Web Control project in Visual Studio to contain code and supporting files for the Mortgage Calculator Web Part.

**To create a Web control project to contain the mortgage calculator Web Part**

1. Start Visual Studio 2005.
2. On the **File** menu, point to **New**, and then click **Project**.

The **New Project** dialog box appears.

1. In the **Project Type** pane, in the **Visual C#** node, select **Windows**.
2. In the **Templates** pane, click **Web Control Library**.
3. Rename your project from **WebControlLibrary1** to **XlMortgageCalc**.

**Setting Up Your Project**

Before you can start working with the code for a Web Part, you must make the following changes to your project settings:

|  |
| --- |
| **NoteNote:** |
| You must make these changes for any Web Part that you create, so you might want to keep this information available for future use. |

* Add the necessary references to create a SharePoint Web Part.
* Remove unnecessary code from the project template.
* Set the version number of the Web Part assembly.
* Sign the Web Part assembly with a strong name.

**Adding Necessary References to Create a SharePoint Web Part**

A Web Part that is used on a SharePoint page must inherit from **Microsoft.SharePoint.WebPartPages.WebPart**. It also uses the **System.Xml.Serialization** namespace to declare the attributes of the Web Part. Therefore, you must add references to those libraries, as follows.

**To add references to the project**

1. On the **Project** menu, click **Add Reference**.
2. In the **Add Reference** dialog box, on the **.NET** tab, select **Microsoft.SharePoint**.

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

1. Click **OK**.

|  |
| --- |
| **NoteNote:** |
| The previous steps assume that you are building the Web Part on a computer where Office SharePoint Server 2007 is installed. You can also find Microsoft.SharePoint.dll at *drive:*\Program Files\Common Files\Microsoft Shared\Web server extensions\12\ISAPI. |

1. Repeat steps 1 to 3 to add a reference to **System.Xml**.

**Removing Unnecessary Code from the Project Template**

The WebCustomControl1.cs file contains code that is created by default when you create the Web Control project.

**To remove unnecessary code from the project template**

1. Rename the WebCustomControl1.cs file to **MortgageCalcWebPart.cs**.
2. In the MortgageCalcWebPart.cs class, replace all of the code that is inserted in the file by default with the following:

C#

[[http://msdn2.microsoft.com/msdn/Controls/CodeSnippet/en-us/copy_off.gif](javascript:CopyCode('ctl00_LibFrame_ctl05CSharp');)Copy Code](javascript:CopyCode('ctl00_LibFrame_ctl05CSharp');)

using Microsoft.SharePoint.WebPartPages; using System; using System.Collections.Generic; using System.ComponentModel; using System.Text; using System.Web; using System.Web.UI; using System.Web.UI.HtmlControls; using System.Web.UI.WebControls; using System.Xml.Serialization; namespace XlMortgageCalc { [XmlRoot(Namespace = "XlMortgageCalc")] public class XlMortgageCalc : WebPart { protected override void RenderWebPart(HtmlTextWriter output) { base.RenderWebPart(output); } } }

**Setting the Version Number for the Project**

By default, the **AssemblyVersion** property of your project is set to increment each time you recompile your Web Part. A Web Part Page identifies a Web Part with the version number that is specified in the web.config file.

With the **AssemblyVersion** property set to increment, if you recompile your Web Part after importing it into a Web Part Page, the Web Part Framework looks for the version number that you specified in the web.config file. If the version number does not match, an error occurs. To prevent the version number of your Web Part from incrementing each time you recompile, you must set the version number in AssemblyInfo.cs.

**To set the version number**

1. In Solution Explorer, in the **Properties** node, double-click **AssemblyInfo.cs**.
2. Edit the line [assembly: AssemblyVersion("1.0.\*")], so that it reads [assembly: AssemblyVersion("1.0.0.0")].
3. Save the project.

**Signing the Assembly with a Strong Name**

Web Parts are designed to be distributed over the Internet or an intranet. For security reasons, when you create a custom Web Part, you must sign it with a strong name to ensure that the Web Part can be trusted by your users.

A strong name consists of the assembly's identity, a public key, a digital signature, and so on. You can use the strong name tool (sn.exe) to manage keys, generate signatures, and verify signatures. This tool is installed with the Microsoft .NET Framework software development kit (SDK) when you install Visual Studio 2005.

**To assign a strong name to your assembly**

1. In Solution Explorer, right-click the **XlMortgageCalc** project, and then select **Properties**.
2. On the **Signing** tab, select the **Sign the assembly** check box.
3. In the **Choose a strong name key file** list, select **<New...>**
4. Enter **keypair** as the file name, and clear the **Protect my key file with a password** check box.
5. Save the project.

**Creating the Mortgage Calculator Form**

Now that you have configured your project, you can add the code that displays the mortgage calculator form, create the Excel workbook that performs the mortgage calculation, and add the code that calls Excel Web Services to load the workbook and perform the mortgage calculation.

**Building the Mortgage Calculation Form**

The mortgage calculation form will contain three input items and one output, as follows:

* Principal amount (input)
* Interest rate (input)
* Loan length (input)
* Monthly payment (output)

To create your form, first declare the members of the Web Part that you will display:

C#

[[http://msdn2.microsoft.com/msdn/Controls/CodeSnippet/en-us/copy_off.gif](javascript:CopyCode('ctl00_LibFrame_ctl06CSharp');)Copy Code](javascript:CopyCode('ctl00_LibFrame_ctl06CSharp');)

protected TextBox \_txtPrincipalAmount = BuildTextBox("amount"); protected TextBox \_txtMortgageLength = BuildTextBox("length"); protected TextBox \_txtInterestRate = BuildTextBox("rate"); protected Button \_cmdCalculate = new Button(); protected Label \_lblTotal = new Label(); protected Label \_lblError = new Label();

Next, add the following methods to build up the layout of the Web Part:

C#

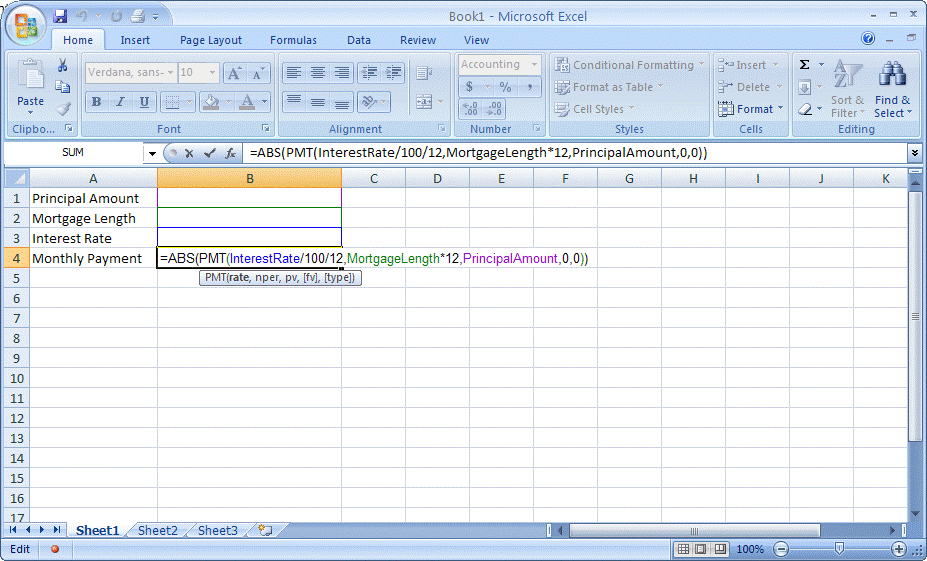
[[http://msdn2.microsoft.com/msdn/Controls/CodeSnippet/en-us/copy_off.gif](javascript:CopyCode('ctl00_LibFrame_ctl07CSharp');)Copy Code](javascript:CopyCode('ctl00_LibFrame_ctl07CSharp');)

/// Override the Init, to attach events and so on. /// </summary> protected override void OnInit(EventArgs e) { this.Load += new EventHandler(MortgageCalcPart\_Load); base.OnInit(e); } /// Handle the load event /// and build the user interface for the mortgage calculator. private void MortgageCalcPart\_Load(object sender, EventArgs e) { // Build up the table that is our user interface. Table t = new Table(); TableRow trMortgageAmount = BuildTableRow(); TableRow trMortgageLength = BuildTableRow(); TableRow trInterestRate = BuildTableRow(); TableRow trCalculate = BuildTableRow(); TableRow trPayment = BuildTableRow(); trMortgageAmount.Cells[0].Text = "Principal Amount"; trMortgageAmount.Cells[1].Controls.Add(\_txtPrincipalAmount); trMortgageAmount.Cells[2].Text = "dollars"; trMortgageLength.Cells[0].Text = "Mortgage Length"; trMortgageLength.Cells[1].Controls.Add(\_txtMortgageLength); trMortgageLength.Cells[2].Text = "years"; trInterestRate.Cells[0].Text = "Interest Rate"; trInterestRate.Cells[1].Controls.Add(\_txtInterestRate); trInterestRate.Cells[2].Text = "%"; cmdCalculate.Text = "Calculate"; trCalculate.Cells[1].Controls.Add(\_cmdCalculate); // \_cmdCalculate.Click += new EventHandler(CalculateUsingWebService); trPayment.Cells[0].Text = "Total Payment"; \_lblTotal.CssClass = "total"; trPayment.Cells[1].Controls.Add(\_lblTotal); trPayment.Cells[2].Text = "per month"; t.Rows.AddRange(new TableRow[]{ trMortgageAmount, trMortgageLength, trInterestRate, BuildTableRow(), trCalculate, BuildTableRow(), trPayment }); Table t2 = new Table(); t2.CssClass = "mainTable"; t2.Width = new Unit(100, UnitType.Percentage); t2.Height = new Unit(100, UnitType.Percentage); t2.Rows.Add(new TableRow()); t2.Rows[0].Cells.Add(new TableCell()); t2.Rows[0].Cells[0].VerticalAlign = VerticalAlign.Middle; t2.Rows[0].Cells[0].HorizontalAlign = HorizontalAlign.Center; t2.Rows[0].Cells[0].Controls.Add(t); this.Controls.Add(t2); } /// Build a new row for the table. /// </summary> /// <remarks>The row has 3 cells</remarks> /// <returns>A TableRow that contains 3 cells</returns> private static TableRow BuildTableRow() { TableRow r = new TableRow(); r.Cells.Add(new TableCell()); r.Cells.Add(new TableCell()); r.Cells.Add(new TableCell()); r.Cells[1].HorizontalAlign = HorizontalAlign.Right; return r; } /// Build a new textbox that has standard props. /// </summary> /// <param name="id">The id for the textbox.</param> /// <returns>A TextBox</returns> private static TextBox BuildTextBox(string id) { TextBox t = new TextBox(); t.ID = id; t.CssClass = "textInput"; return t; }

**Creating the Excel Workbook That Performs the Mortgage Calculation**

The next step in the project takes you out of Visual Studio 2005 and into Excel 2007 to create the workbook that will be used by your Web Part, as shown in Figure 2.

**Figure 2. Excel 2007 workbook that performs mortgage calculation**



The workbook itself is simple, but it demonstrates an important point: You can now integrate business logic from Excel with other line-of-business applications. By doing so, you can reduce costs because business analysts can make changes to the calculation model without requiring developers to recode a solution.

**To create the workbook**

1. Start Excel 2007.
2. Enter the following information into the first worksheet, "Sheet1":

|  |  |  |
| --- | --- | --- |
|  | A | B |
| 1 | Principal Amount |  |
| 2 | Mortgage Length |  |
| 3 | Interest Rate |  |
| 4 | Monthly Payment |  |

1. Name the following cells to make them easier to reference.
   * Name cell B1 **PrincipalAmount**.
   * Name cell B2 **MortgageLength**.
   * Name cell B3 **InterestRate**.
   * Name cell B4 **Payment**.

|  |
| --- |
| **NoteNote:** |
| To name a cell, click **Name a Range** on the **Formulas** tab. |

1. Next, in cell B4, enter the following formula: **=ABS(PMT(InterestRate/100/12,MortgageLength\*12,PrincipalAmount,0,0))**
2. Finally, on the Ribbon, on the **Home** tab, change the format of cell B4 to **currency** by using the formatting options.
3. Now that your workbook is finished, use the **Save As** option in Excel to save to **Excel Services**. Save your workbook to a document library on Office SharePoint Server 2007 that is configured as a trusted location for Excel Services. For this article, the workbook is saved to: http://YourServer/Documents/MortgageCalc.xlsx.

|  |
| --- |
| **NoteNote:** |
| For more information about how to save a workbook from Excel to a document library, see [How to: Save from Excel Client to the Server](http://msdn2.microsoft.com/en-us/library/ms494032.aspx) in the Microsoft Office SharePoint Server 2007 SDK.  For information about how to configure a document library as a trusted location, see [How to: Trust a Location](http://msdn2.microsoft.com/en-us/library/ms517291.aspx). |

**Calling the Excel Web Service to Perform the Mortgage Calculation**

Now that you have created the form for mortgage calculation and the workbook that will perform the actual calculation, it is time to add the code to your Web Part that calls Excel Web Services to load the workbook, set the input values, and retrieve the output value for your mortgage calculator.

The steps for doing this are as follows:

1. Add a reference to the Excel Web Services library.
2. Add code to your Web Part to call the Excel Web Services DLL (Microsoft.Office.Excel.Server.WebServices.dll).

|  |
| --- |
| **NoteNote:** |
| You link directly to the Excel Web Services library because you are writing a Web Part that is executed within the Microsoft Windows SharePoint Services environment, rather than calling the Web service methods by using SOAP over HTTP. By linking directly in this scenario, you gain better performance and security, and you are guaranteed that your mortgage calculation form will work in all SharePoint topologies. If you were building a separate Web application (a Web application not running within Windows SharePoint Services), you would want to use the true Web service interface.  For more information about when to link directly to the Excel Web Services DLL and when to call Excel Web Services by using SOAP over HTTP, see [Loop-back SOAP Calls and Direct Linking](http://msdn2.microsoft.com/en-us/library/ms565591.aspx) in the Microsoft Office SharePoint Server 2007 SDK. |

**Adding a Reference to the Excel Web Service**

To add a reference to Excel Web Services, you must have a server that has Office SharePoint Server 2007 installed. You add a reference to the Excel Web Services DLL (Microsoft.Office.Excel.Server.WebServices.dll) just as you did with the Microsoft.SharePoint.dll earlier.

**To add a reference to Microsoft.Office.Excel.Server.WebServices.dll**

1. On the **Project** menu, click **Add Reference**.
2. In the **Add Reference** dialog box, on the **.NET** tab, select **Microsoft.Office.Excel.Server.WebServices.dll**.

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

1. Click **OK**.

|  |
| --- |
| **NoteNote:** |
| The previous steps assume that you are building the Web Part on a computer that has Office SharePoint Server 2007 installed. You can also find Microsoft.Office.Excel.Server.WebServices.dll at *drive:*\Program Files\Common Files\Microsoft Shared\Web server extensions\12\ISAPI. |

1. Finally, at the top of your MortgageCalcWebPart.cs file, add the following **using** directive:

[[http://msdn2.microsoft.com/msdn/Controls/CodeSnippet/en-us/copy_off.gif](javascript:CopyCode('ctl00_LibFrame_ctl11other');)Copy Code](javascript:CopyCode('ctl00_LibFrame_ctl11other');)

using Microsoft.Office.Excel.Server.WebServices;

**Calling the Excel Web Service**

With the reference added to Excel Web Services, you can now call to load the workbook that you created earlier, set the input values, and retrieve the output value to display in your form. To do that, make the following changes to your MortgageCalcWebPart.cs file:

1. In the **MortgageCalcPart\_Load** method, uncomment the following line:

[[http://msdn2.microsoft.com/msdn/Controls/CodeSnippet/en-us/copy_off.gif](javascript:CopyCode('ctl00_LibFrame_ctl12other');)Copy Code](javascript:CopyCode('ctl00_LibFrame_ctl12other');)

\_cmdCalculate.Click += new EventHandler(CalculateUsingWebService);

1. Add the following code to the **MortgageCalcWebPart** class:

C#

[[http://msdn2.microsoft.com/msdn/Controls/CodeSnippet/en-us/copy_off.gif](javascript:CopyCode('ctl00_LibFrame_ctl13CSharp');)Copy Code](javascript:CopyCode('ctl00_LibFrame_ctl13CSharp');)

/// Calculate the workbook by calling Excel Web Services. /// </summary> private void CalculateUsingWebService(object sender, EventArgs e) { Status[] status; string sessionId = null; // Initialize the Excel Web Service. ExcelService es = new ExcelService(); // Open the workbook – this loads the workbook // that was saved to the SharePoint document library // and returns a sessionId that can be used // to perform further operations on the workbook. try { sessionId = es.OpenWorkbook("http://TODORenameToYourServer/Documents/MortgageCalc.xlsx", "en-US", "en-US", out status); } catch (Exception ex) { sessionId = null; } // Ensure that the workbook has been // successfully opened on the server. // If not, show an error message to the user. if (sessionId == null) { ShowError("Error opening workbook. Check the URL in OpenWorkbook, and be sure that the workbook is in a Trusted Location"); return; } // Set the input values from the form into the workbook // that was loaded on Excel Services. // These values are used to calculate // the monthly mortgage payment. es.SetCellA1(sessionId, "Sheet1", "PrincipalAmount", \_txtPrincipalAmount.Text.Trim(), out status); es.SetCellA1(sessionId, "Sheet1", "MortgageLength", \_ txtMortgageLength.Text.Trim(), out status); es.SetCellA1(sessionId, "Sheet1", "InterestRate", \_txtInterestRate.Text.Trim(), out status); // Calculate the workbook now that the new values are set. // This is not necessary because // the spreadsheet is in 'automatic' calculation mode, // but it is good practice, and good demonstration. es.CalculateWorkbook(sessionId, CalculateType.Recalculate, out status); // Retrieve the result of the workbook calculation // from Excel Services and display it in the form. // Or, if there is a problem getting the value, // show an error to the user. object o = es.GetCellA1(sessionId, "Sheet1", "Payment", true, out status); if (o != null) { \_lblTotal.Text = Convert.ToString(o); } else { ShowError("Error getting total value from workbook."); return; } } /// Show an error message to the user, and remove all other controls from the Web Part. /// </summary> /// <param name="message">The message to display</param> private void ShowError(string message) { \_lblError.Text = message; \_lblError.CssClass = "error"; this.Controls.Clear(); this.Controls.Add(\_lblError); }

1. Save and build your solution to ensure that there are no errors.

You created a Web Part that calls Excel Services (by using the Excel Web Services interface) to load, set values, calculate, and retrieve values from a workbook saved to Office SharePoint Server 2007. Now, all you need to do is package and deploy the Web Part to Office SharePoint Server 2007.

**Conclusion**

This article describes how to create a Web Part for Office SharePoint Server 2007 that calls Excel Web Services to perform a calculation. The next step is to create a deployment project to deploy and use the Web Part. To learn how to package and deploy the Web Part that you created, and how to improve the appearance and configurability of a Web Part, see [Deploying and Optimizing a SharePoint Web Part That Calls Excel Web Services](http://msdn2.microsoft.com/en-us/library/aa973805.aspx).

**About the Author**

Dan Battagin is a lead program manager on the Excel team who also worked on Excel Services