Lab 17: Excel Services and Report Center

**Lab Time**: 45 Minutes

**Lab Directory**: C:\Student\Labs\17\_ExcelServices

**Lab Overview**: One of the difficulties of working with Excel workbook data is sharing the information contained in the workbook without multiple versions floating around. It is quite common to have an updated workbook created, but to still have users working on a previous version of the workbook, which of course creates incorrect data. This can be a major problem when the workbook is being used for mission critical calculations. In this lab you will explore the new Excel Services capability of MOSS to interact with data stored in workbooks on the server. By interacting with a server-based workbook you can prevent versioning issues and allow for greater accessibility to the data. Furthermore, you can utilize various features found in MOSS to ease the user interaction without requiring complex custom code.

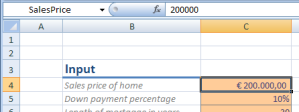
In the first exercise, you will examine a mortgage calculation workbook which will be published to a SharePoint document library and driven through Excel Services. You will interact with the workbook on the server by providing new data for fields and calculations. In the second exercise, you will examine how pivot tables and charts behave when deployed on the server. The third exercise involves the addition of custom code to drive the mortgage calculator. This will allow you to access logic and data in back-end systems from inside a workbook.

# Exercise 1: Publishing Excel Workbooks with Excel Services

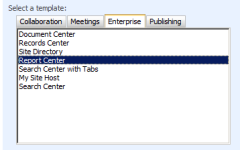
In this exercise you will examine a prebuilt mortgage calculation workbook inside Excel. Next, you will deploy this workbook to a SharePoint **Report Library** and examine the behavior from inside a web-browser. Before deploying the workbook, the SharePoint environment needs to be setup. You will create a new site collection with a top-level site based on the **Report Center** site template. Next you will configure a **Report Library** as a **Trusted Location** to allow access to the workbook reports through Excel Services.

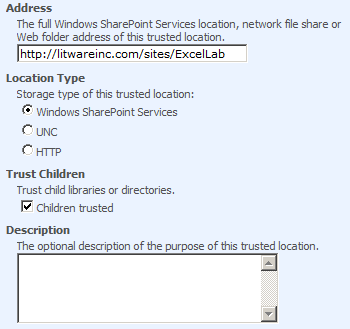
The **Report Center** template contains facilities for creating dashboard pages, **Key Performance Indicator** overviews, as well as providing a library type for reports. Trusted locations are used to ensure data integrity and quality as well as providing a centralized environment for managing resources and user access.

1. Start by examining the mortgage calculation workbook in Excel 2007. The mortgage calculation workbook is a simple workbook using the calculation engine of Excel to provide insight into the height of monthly mortgage payments based on house price and other variables.
   1. Open Windows Explorer and navigate to the following location: **C:\Student\Labs\17\_ExcelServices\Starter Files**
   2. Open the **Mortage Calculator.xlsx** workbook in Excel. The workbook contains two worksheets. The **Mortgage Calculator** worksheet is used for calculating mortgages. The second worksheet contains private instructions which should not be made publicly available.
   3. Examine the **Mortage Calculator** worksheet. Alter the values for the mortgage calculation using the following information.
      1. Sales price is **$200,000**
      2. Down payment is **10%**
      3. Length of mortgage is **30 years**
      4. Annual interest is **6.5%**
   4. Check that the calculated monthly payment comes down to **$1,137.72.**
   5. Take notice of the fields where you entered data. Besides being identified by a column / row index, the cells also have a name defined for them. This will allow you to specify values for these cells through Excel Services.

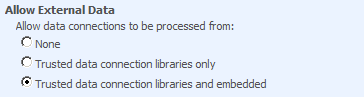


* 1. Examine the **Mortgage Guidelines** worksheet. It contains text which should remain outside of the public domain.
  2. Keep **Excel** open, as you will return to it after finishing the next step.

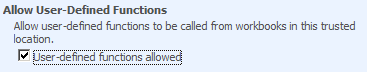
1. Next, you will create a new site collection with a top-level site based on the **Report Center** site template. The Report Center site will be used to store workbook reports and provides other facilities common to a reporting environment. In our case, the site will be used to host the mortgage calculation workbook and the other workbooks used in later exercises.
   1. Open **SharePoint 3.0 Central Administration** and navigate to the **Application Management** page.
   2. On the **Application Management** page, click the **Create Site Collection** link in order to navigate to the Create Site Collection page.
   3. Create a new site collection using the following values.
      1. Be sure that the **Web Application** is set to **http://litwareinc.com/** (Litware Public Site).
      2. Title is **Excel Lab**.
      3. Web Site Address is **http://litwareinc.com/sites/ExcelLab**.  
         
      4. Template Selection is **Report Center**, which you find under the **Enterprise** group.
      5. **Primary Site Collection Administrator** is **LitwareInc\Administrator**.
   4. Click **OK** to start the process of creating the new site collection. When this process has completed successfully you will be presented with an administration page that links to the new top-level site.
2. Now it's time to configure the URL for the Report Center site as a Trusted Location so that it's possible to publish workbooks there that will be loaded and rendered by Excel Services. This is a crucial step because Excel Services only works with workbooks it finds in trusted locations to manage access to resources.
   1. Open **SharePoint 3.0 Central Administration** and navigate to the **Application Management** page.
   2. Click **Litware SSP (Default)** on the **Quick Launch** in order to navigate to the shared services administration page.  
      On the **Shared Services Administration** page, click **Trusted file locations** in order to manage trusted file locations for Excel Services.
   3. Click **Add Trusted File Location** to add the location of the Report Center site you created in the previous step of this exercise. The **Location** section is the first section on the page. Create the trusted file location using the following values.
      1. **Address** is **http://litwareinc.com/sites/ExcelLab**.
      2. **Location Type** is **Windows SharePoint Services**.
      3. **Trust Children** is checked



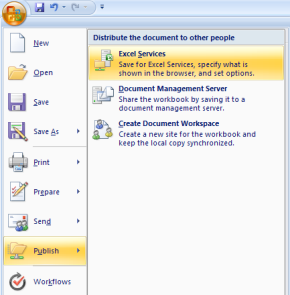
* + 1. **Scroll down and locate the External Data section. Allow External Data** is **Trusted data connection libraries and embedded**.

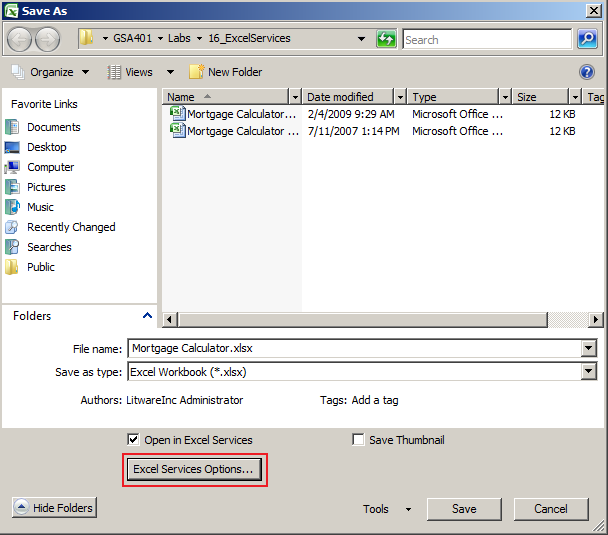


* + 1. **The last section is the User-Defined Functions section. Make sure Allow User-Defined Functions** is checked.

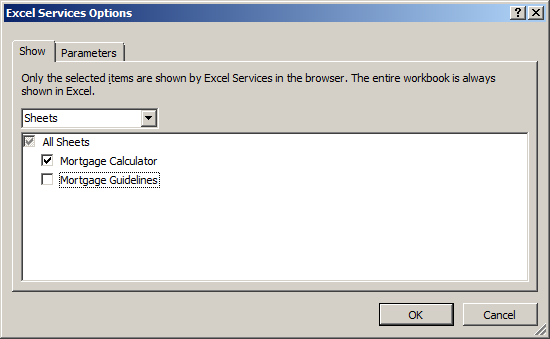
  
NOTE: Trusting children automatically trusts all document libraries in the new Report Center site. The external data will be used in the next exercise to access data from a LOB database. The User Defined Function is a feature that will be used in the last exercise to drive the mortgage calculation using code in a .NET assembly.

* 1. Press **OK** to add the new file location to the list of trusted file locations.
  2. Open a new command window and issue the **IISRESET** command to reset the web server and allow the new trusted location to become immediately available.

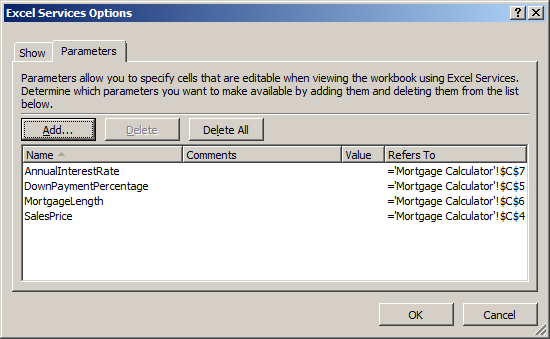
1. Now it's time to publish the mortgage calculation workbook. The mortgage calculation workbook needs to be made available online to allow users to interact with the calculator without having Excel installed on their desktop, as well as allowing users to always work with the latest mortgage calculation formula.
   1. Go back to the **Mortgage Calculator.xlsx** which should have been left open in Excel from the previous exercise.
   2. On the **Office** Menu, click **Publish > Excel Services** to publish the workbook to SharePoint.
   3. In the **Save As** window, click the **Excel Services Options** button to configure the workbook before publishing it to Excel Services.



* + 1. On the **Show** tab, choose **Sheets** to only allow certain sheets to be made available.
    2. Make sure that only the **Mortgage Calculator** worksheet is published by clearing the checkbox for all other worksheets.



* + 1. On the **Parameters** tab, review the parameters that are made available for modification through Excel Services. If all 4 of the parameters are not already in the listbox shown (**AnnualInterestRate**, **DownPaymentPercentage**, **MortgageLength**, and **SalesPrice**), then click the **Add** button, place a check next to any of the parameters that were not included already, and click **OK**.



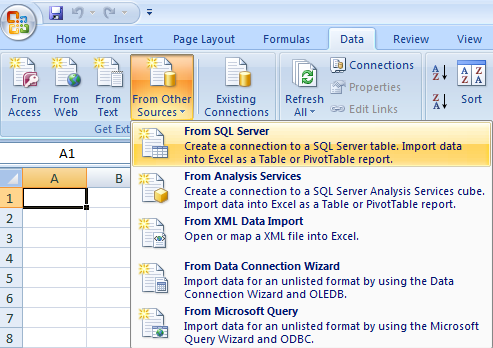
* + 1. Click **OK** to close the **Excel Services Options** dialog.
  1. In the **Save As** window, Click on **My Network Places** and double click on **ReportsLibrary on litwareinc.com**, then type **Mortgage Calculator.xlsx** into the **File Name:** textbox. If **ReportsLibrary on litwareinc.com** is not already there, then you need to make the connection available by doing the following [If it is there already, then click **SAVE** and move on to step e.]:
     1. Type or paste this link into the **File Name:** textbox: **http://litwareinc.com/sites/ExcelLab/ReportsLibrary/Mortgage Calculator.xlsx.**
     2. Click **SAVE.**
  2. In the **Choose Document Type** dialog, choose **Report** and click **OK** to publish the workbook.
  3. The workbook is automatically opened in a web-browser. Keep this browser window open for the next exercise.

1. Finally, examine the mortgage calculation workbook in Excel Services. The view of the mortgage calculator workbook is nearly the same as the Excel-based view. The users can now interact with the workbook by providing values for the named cells. These values are entered in a different fashion than the Excel counterpart. Percentage values are a sample of this.
   1. Examine the **Mortage Calculator** worksheet using Excel Services. Alter the values for the mortgage calculation using the following information. Take note of the decimal separator character. It is locale specific and depends on the locale configured in SharePoint. Also take note of how percentages are noted, using the value '1' for 100%.
      1. Sales price is **200,000**
      2. Down payment is **0.1**
      3. Length of mortgage is **30**
      4. Annual interest is **0.065**
   2. Check that the calculated monthly payment comes down to **$1,137.72.**

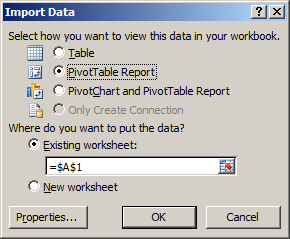
# Exercise 2: Publishing Excel Workbooks with a Data Connection

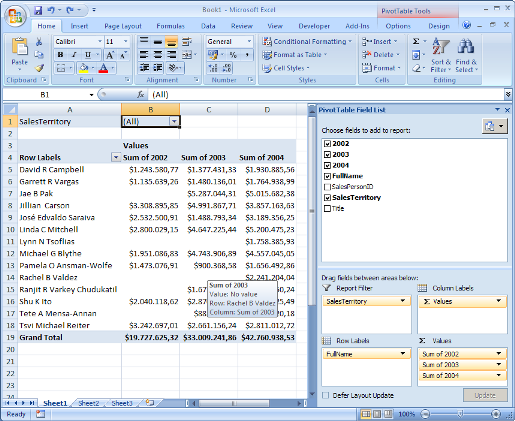
While the ability to publish a workbook online poses a great solution platform, many workbooks are not a self-contained unit. Instead they retrieve information from an array of external data sources such as the Microsoft SQL Server database engine and Analysis Services. In this exercise you will create a new workbook which rolls up information about sales based on the region, salesperson and year. Next the data will be displayed in a pivot-chart before publishing the spreadsheet to Excel Services.

1. In this step you will create the sales overview using a pivot table. First, you will create a new workbook which will display a sales overview in a pivot table. Next, you will create a pivot chart that is defined on top of this table.
   1. Open a new instance of **Excel**. An empty workbook will be made available automatically. [**IMPORTANT**: Don’t just open a new workbook in the same instance of Excel that you have been using. You must close and reopen Excel for publishing to work correctly later.]
   2. On the **Get External Data** group in the **Data** tab, click **From Other Sources** and then **From** **SQL Server** in order to create a new data connection to SQL Server.



* 1. In the **Data Connection Wizard** dialog perform the following steps.
     1. In the **Connect to Database Server** step, enter **LITWARESERVER** for the server name. Leave the log on credentials to **Windows Authentication**. Click **Next** to navigate to the next step.
     2. In the **Select Database and Table** step, select **AdventureWorks2008** as the database and **vSalesPersonSalesByFiscalYears** view as the data source. Click **Finish** to complete the wizard and close the dialog.
  2. In the **Import Data** dialog, select **PivotTable Report** and click **OK** to add a new pivot table bound to the data source to the first worksheet.



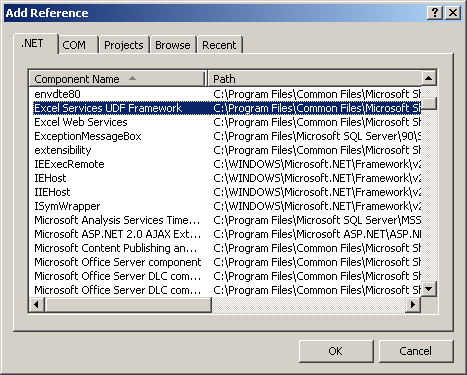
* 1. In the **Pivot Table Field List** pane, configure the pivot table according to the following information. If the pane does not appear, right-click anywhere in the pivot table and choose **Show field list**. Drag the fields from the field list into the four container areas at the bottom of the pane.
     1. For **Row Labels**, add the **FullName** field
     2. For **Values**, add the **2002**, **2003** and **2004** fields. Configure each field to display the SUM instead of the default COUNT.
        1. Click the field and choose **Value Fields Settings…**.
        2. On the **Summarize by** tab, choose **Sum**.
        3. Optionally change the number format to reflect the **Currency** type.
        4. Click **OK** to close the settings dialog.
     3. For **Report Filter**, add the **SalesTerritory** field.
     4. The following image depicts the configured pivot table.
     5. Select a cell in the pivot table. The next step will create a new pivot chart, and when doing so Excel will recognize that the chart needs to be bound to the selected pivot table. In the sample image above cell **B1** is selected, containing the report filter.
     6. Right-click the **Sheet1** tab, while keeping the cell selected. The tab can be found on the bottom of the Excel window. Choose **Insert** in order to open the sheet creation dialog.
     7. In the **Insert** dialog, choose **Chart** and click **OK** to insert a new chart sheet into the workbook. The chart should automatically display the records of the pivot table.
     8. Inside the **Type** group, on the **Pivot Chart Tools – Design tab**, click **Change Chart Type** to alter the chart type. Choose the **Clustered Cylinder** chart type.
     9. Optionally provide a value for the **SalesTerritory** filter to lessen the number of records displayed. The following image depicts how the chart can look when filtering on the **Northwest** Territory and altering the axis labels. 
     10. On the **Office** menu, click **Save** in order to open the Save dialog. Choose **Sales Overview.xlsx** as the name for the document. Save this workbook to **C:\Student\Labs\17\_ExcelServices\Lab**.
     11. Keep the workbook open as you will return to it in the following task.

1. Now it's time to publish the sales overview workbook to allow users to see a daily overview of the data in the sales workbook.
   1. Go back to the **Sales Overview.xlsx** that should still be open in Excel.
   2. On the **Office** Menu, click **Publish** **>** **Excel Services** to publish the workbook to SharePoint.
   3. In the **Save As** dialog, click **Excel Services Options** to configure the workbook before publishing it to Excel Services.
      1. On the **Show** tab, choose **Sheets** to only allow certain sheets to be made available.
      2. Make sure that only the sheet containing the table and the one containing the chart are published by clearing the checkbox for all other worksheets.
      3. Choose **OK** to close the Excel Services Options dialog.
   4. In the **Save As** window, Click on **My Network Places** and double click on **ReportsLibrary on litwareinc.com**, then type **Sales Overview.xlsx** into the **File Name:** textbox. Click **SAVE**.
   5. In the **Choose Document Type** dialog pop-up, choose **Report** and click **OK** to publish the workbook.
   6. The workbook is automatically opened in a web-browser.
2. Finally, examine the mortgage calculation workbook in Excel Services. You should observe that the view of the sales overview workbook is nearly the same on the web as in Excel. 3D charts are converted to their 2D equivalent and are rendered using full HTML.
   1. Examine the worksheet containing the pivot table in Excel Services. Alter the value for the **SalesTerritory** filter.
   2. Examine the chart sheet containing the pivot chart in Excel Services. Notice that the view is similar to the view provided by Excel, but differing slightly in layout.

# Exercise 3: Calling a method from a .NET Assembly from a Server-side Workbook

While the calculation engine provided by Excel is very elaborate and powerful, there might be times where there is a need to extend the functionality provided by Excel or to access previously created business logic. For standard client-side workbooks loaded inside a desktop version of Excel, one would resort to using a tool such as Visual Studio Tools for Office to embed custom .NET code right inside the workbook. A workbook hosted by Excel Services is not allowed to use these extensions since there is no client application to host the add-in runtime. Excel Services does allow a workbook to call into back-end code by using a User Defined Function (UDF). Using UDFs you can call a method on a class and return values back into the workbook. You can use values, cells, or cell ranges as the input for the UDF and also retrieve a similar set of values from the return value.

In this exercise you will move the mortgage calculation function of the first exercise into a new code class. This will allow the mortgage calculation to be shared by many different applications requiring mortgage calculation. The workbook will use a User Defined Function to call into the code class to retrieve the mortgage estimate.

1. Create a Visual Studio project to develop a new UDF class library. Before the workbook can access the extracted function, a new class library needs to be created and deployed into a location known to Excel Services.
   1. Open **Microsoft Visual Studio 2008** and create a new **Class Library** project named **MortgageCalculator**. Be sure to select **.NET Framework** **3.0** in the dropdown box in the top right-hand corner. This is because .NET 3.0 is automatically included in MOSS 2007; however, .NET 3.5 is chosen by default in Visual Studio 2008, but may not be installed as a general rule in a company’s farm. Create the project in the **C:\Student\Labs\17\_ExcelServices** directory.
   2. In the Solution Explorer, right-click the **Class1.cs** node and click **Rename**. Name the file **Calculator.cs**.
   3. In the Solution Explorer, right-click the **References** node and choose **Add Reference** in order to open the **Add Reference** dialog.
   4. In the **Add Reference** dialog, choose the assembly with the component name of **Excel Services UDF Framework** to add a reference to **Microsoft.Office.Excel.Server.Udf.dll** and then click **OK**.  
      
   5. In the **Solution Explorer**, double-click the **Calculator.cs** node in order to open the code file in the Visual Studio editor.
   6. In the **Calculator.cs** code file, perform the following steps.
      1. Add a new public method to the **Calculator** class called **CalculateMortgage**. This method should take four parameters and return a **double** value.
         1. **SalesPrice** of type **int**.
         2. **MortgageLength** of type **int**.
         3. **DownPaymentPercentage** of type **double**.
         4. **AnnualInterestPercentage** of type **double**.
      2. Hard-code a return value of -1, in order for the project to compile correctly. Later on you will provide a real implementation for the mortgage calculation method.
      3. Add a new **using** statement that includes the **Microsoft.Office.Excel.Server.Udf** namespace. This namespace is part of the **Microsoft.Office.Excel.Server.Udf.dll** assembly referenced earlier.
      4. Adorn the **Calculator** class with the **[UdfClass]** attribute.
      5. Adorn the **CalculateMortgage** method with the **[UdfMethod]** attribute.
      6. The final class definition should look like the following code sample.

using System;

using Microsoft.Office.Excel.Server.Udf;

namespace MortgageCalculator {

[UdfClass]

public class Calculator {

[UdfMethod]

public double CalculateMortgage(int salesPrice,

int mortgageLength,

double downPaymentPercentage,

double annualInterestPercentage) {

// implementation

return -1;

}

}

}

* 1. Compile the project by clicking **Build MortgageCalculator** on the **Build** menu. Make sure that the project compiles correctly. Repair any errors that might turn up.
  2. In the **CalculateMortgage** method, provide an implementation for the calculation. The following sample displays a similar calculation as was provided earlier in the **Mortgage Calculator.xlsx** workbook.

double financed = (1 - downPaymentPercentage) \* salesPrice;

int nrOfMonths = mortgageLength \* 12;

double monthlyInterestRate = annualInterestPercentage / 12;

return financed \* (monthlyInterestRate / (1 - Math.Pow((1 + monthlyInterestRate), nrOfMonths \* -1)));

* 1. Compile the project by clicking **Build MortgageCalculator** on the **Build** menu. Make sure that the project compiles correctly. Repair any errors that might turn up.

1. Now it's time to deploy the assembly DLL produced by your UDF class library project. The compiled UDF class needs to be placed in a location registered in Excel Services. You can register arbitrary locations on the file-system or use the Global Assembly Cache. In this lab, you will use a common UDF directory which will be registered with Excel Services.
   1. Open Windows Explorer and navigate to the root folder of the **C:\** drive.
   2. Add a new folder named **C:\UDFs**. Since the files in this directory will be accessed by the IIS worker process for the hosting Web Application, the service account tied to this worker process, which will be servicing Excel Services requests, needs read access to this folder.
   3. Right-click the directory and choose **Properties**.
   4. On the **Security** tab, add the **litwareinc\sys-spWorkerProcess** account and give this account read access by doing the following:
      1. Click **Add**.
      2. Type **litwareinc\sys-spWorkerProcess** in the **Enter the object names to select (examples)** textbox.
      3. Click **OK**.
      4. Make sure **litwareinc\sys-spWorkerProcess (sys-spWorkerProcess@litwareinc.com)** is highlighted.
      5. Check **Read** in the **Permissions for litwareinc\sys-spWorkerProcess** box.
      6. Click **OK**.
   5. Copy the **MortgageCalculator.dll** assembly into the new UDFs folder. The **MortgageCalculator.dll** assembly can be found in your project directory at the following path **\MortgageCalculator\bin\Debug**.
2. Register the UDF class library with Excel Services. The new location of the UDF assembly needs to be trusted by Excel Services for the call into the UDF to work.
   1. Open **SharePoint 3.0 Central Administration.**
   2. On the **Quick Launch** select the **LitwareSSP** Shared Services Provider.
   3. On the **Shared Services Administration** page, click **User-defined function assemblies** in order to manage UDF assemblies registered with Excel Services.
   4. On the **User-Defined Functions** page click **Add User-Defined Function Assembly** in order to register a new UDF assembly with Excel Services.
   5. On the **Add User-Defined Function Assembly** page, alter the values for the UDF assembly using the following information.
      1. **Assembly** is **C:\UDFs\MortgageCalculator.dll**.
      2. **Assembly Location** is **File path**.
      3. Click **OK** to add the assembly to the registered UDF assemblies.
3. Modify an Excel workbook to call the UDF method. This time the workbook itself will not contain the implementation for calculating a mortgage. Instead, the implementation will be leveraged from the UDF in the .NET you created in the previous steps.
   1. Open Windows Explorer and navigate to the following location: **\Student\Labs\17\_ExcelServices\Starter Files**.
   2. Open the **Mortage Calculator UDF.xlsx** workbook in Excel. The workbook is similar to the earlier mortgage calculation form, only without the calculation logic.
   3. In cell C9, create a function that calls the **CalculateMortgage** function defined earlier in the class library. Use the defined names for the four value cells to pass parameters into the **CalculateMortgage** function. A sample of the function to use is as follows:

=CalculateMortgage(SalesPrice,MortgageLength,DownPaymentPercentage,AnnualInterestRate)

* 1. Keep the workbook open as you will return to it in the next step.

1. Publish the UDF mortgage calculation workbook. This is the last step before you can view the result in Excel Services.
   1. Go back to the **Mortgage Calculator UDF.xlsx** workbook that should still be open in Excel.
   2. On the **Office** Menu, click **Publish** and then **Excel Services** to publish the workbook to SharePoint.
   3. In the **Save As** window, click **Excel Services Options** to configure the workbook before publishing it to Excel Services.
      1. On the **Show** tab, choose **Sheets** to only allow certain sheets to be made available.
      2. Make sure that only the **Mortgage Calculator** worksheet is published by clearing the checkbox for all other worksheets.
      3. On the **Parameters** tab, review the parameters that are made available for modification through Excel Services. If all 4 of the parameters are not already in the listbox shown (**AnnualInterestRate**, **DownPaymentPercentage**, **MortgageLength**, and **SalesPrice**), then click **Add**, place a check next to any of the parameters that were not included already, and click **OK**.
      4. Click **OK** to close the **Excel Services Options** dialog.
   4. In the **Save As** window, type the following URL into the **File name** textbox: **http://litwareinc.com/sites/ExcelLab/ReportsLibrary/Mortgage Calculator UDF.xlsx**.
   5. Click **Save**.
   6. In the **Choose Document Type** dialog, choose **Report** and click **OK** to publish the workbook.
   7. The workbook is automatically opened in a web-browser.
2. Examine the mortgage calculation workbook in Excel Services. The view of the mortgage calculator workbook is nearly the same as the Excel-based view. The users can now interact with the workbook by providing values for the named cells. The calculation takes place on de server inside the custom UDF assembly, instead of through normal Excel functions.
   1. Examine the **Mortage Calculator** worksheet as it appears in Excel Services. Alter the values for the mortgage calculation using the following information. Take note of the decimal separator character. It is locale specific and depends on locale configured in SharePoint. Also take note of how percentages are noted, using the value '1' for 100%.
      1. **Annual interest** is **0.065**
      2. **Down payment** is **0.1**
      3. **Length** of mortgage is **30**
      4. **Sales** price is **200,000**
   2. Verify that the calculated monthly payment comes down to **$1,137.72**