In this article I will explain a tutorial to create a simple Windows Service with sample example in C# and VB.Net.

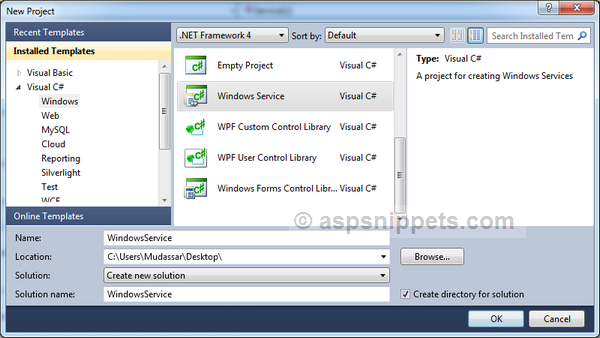
The Windows service will support two modes

1. Interval mode: where Windows Service execute a task at regular intervals after some delay

2. Daily mode: where the Windows Service will execute a task at specific (certain) time of day.

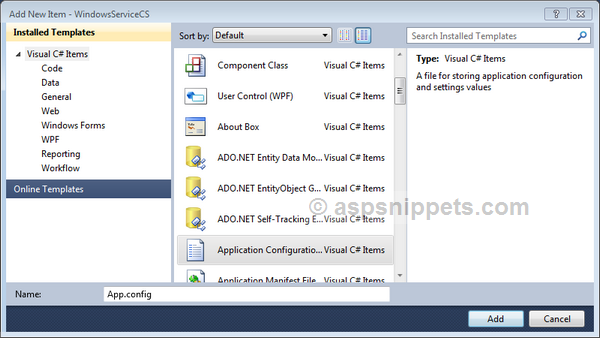
**Creating a Windows Service Project**

The very first step is to add a new project of type Windows Service as shown below.



**Adding App.Config File**

Next you need to add an Application Configuration File (App.Config file). This file will be used to control the Windows Service and make it work in different modes.



After adding the file, you will need to copy the following AppSettings to the App.Config file.

<?xmlversion="1.0"encoding="utf-8" ?>

<configuration>

    <appSettings>

        <add key ="Mode" value ="Daily"/>

        <!-- <add key ="Mode" value ="Interval"/>-->

        <add key ="IntervalMinutes" value ="1"/>

        <add key ="ScheduledTime" value ="18:41"/>

    </appSettings>

</configuration>

Below is the description of each AppSetting.

Mode: It is used to set the Mode. There are two types of modes namely Daily and Interval.

IntervalMinutes: It is used when Mode is set to Interval. It consist of the Interval value in Minutes after which the Windows Service will perform a task. In other words it is the delay value.

ScheduledTime: This setting is used when the Mode is set to Daily. It is used to notify the Windows Service the time it should perform a task. The value specified is in 24 hour time format.

**Timer Configuration**

The following code has to be placed in the Service.cs Class. I am making use of a Timer class belonging to the System.Threading namespace in order to execute the Windows Service periodically at regular intervals and as well as once a day at specific (certain) time of day.

The Timer has a Callback method which gets triggered automatically when the due time is elapsed.

We will start with importing the following namespaces.

**C#**

using System.IO;

using System.Threading;

using System.Configuration;

**VB.Net**

Imports System.IO

Imports System.Threading

Imports System.Configuration

Below is the Windows Service class with the OnStart and OnStop event handlers.

When the Windows Service starts it calls the ScheduleService method which first reads the Mode AppSetting.. There’s a ScheduledTime variable which is set in both modes.

When the Mode is set to Daily then the ScheduledTime is read from the AppSettings. In the case when the scheduled time is passed, it is updated to same time on the next day.

When the Mode is set to Interval then the IntervalMinutes is read from the AppSettings and the schedule time is calculated by adding the IntervalMinutes to the Current Time.

Finally the Timer is set to run the scheduled time. When the scheduled time is elapsed, the Timer’s Callback method is triggered which logs the current date and time to a Text file.

**Note**: While using the above code, you could encounter the following error: ***The type or namespace name 'ConfigurationManager' does not exist in the namespace 'System.Configuration' (are you missing an assembly reference?)***.The solution is provided in my article [The type or namespace name 'ConfigurationManager' does not exist in the namespace 'System.Configuration’](https://www.aspsnippets.com/Articles/The-type-or-namespace-name-ConfigurationManager-does-not-exist-in-the-namespace-SystemConfiguration.aspx).

**C#**

public partial class Service1 : ServiceBase

{

    public Service1()

    {

        InitializeComponent();

    }

    protected override void OnStart(string[] args)

    {

        this.WriteToFile("Simple Service started {0}");

        this.ScheduleService();

    }

    protected override void OnStop()

    {

        this.WriteToFile("Simple Service stopped {0}");

        this.Schedular.Dispose();

    }

    private Timer Schedular;

    public void ScheduleService()

    {

        try

        {

            Schedular = new Timer(new TimerCallback(SchedularCallback));

            string mode = ConfigurationManager.AppSettings["Mode"].ToUpper();

            this.WriteToFile("Simple Service Mode: " + mode + " {0}");

            //Set the Default Time.

            DateTime scheduledTime = DateTime.MinValue;

            if (mode == "DAILY")

            {

                //Get the Scheduled Time from AppSettings.

                scheduledTime = DateTime.Parse(System.Configuration.ConfigurationManager.AppSettings["ScheduledTime"]);

                if (DateTime.Now > scheduledTime)

                {

                    //If Scheduled Time is passed set Schedule for the next day.

                    scheduledTime = scheduledTime.AddDays(1);

                }

            }

            if (mode.ToUpper() == "INTERVAL")

            {

                //Get the Interval in Minutes from AppSettings.

                int intervalMinutes = Convert.ToInt32(ConfigurationManager.AppSettings["IntervalMinutes"]);

                //Set the Scheduled Time by adding the Interval to Current Time.

                scheduledTime = DateTime.Now.AddMinutes(intervalMinutes);

                if (DateTime.Now > scheduledTime)

                {

                    //If Scheduled Time is passed set Schedule for the next Interval.

                    scheduledTime = scheduledTime.AddMinutes(intervalMinutes);

                }

            }

            TimeSpan timeSpan = scheduledTime.Subtract(DateTime.Now);

            string schedule = string.Format("{0} day(s) {1} hour(s) {2} minute(s) {3} seconds(s)", timeSpan.Days, timeSpan.Hours, timeSpan.Minutes, timeSpan.Seconds);

            this.WriteToFile("Simple Service scheduled to run after: " + schedule + " {0}");

            //Get the difference in Minutes between the Scheduled and Current Time.

            int dueTime = Convert.ToInt32(timeSpan.TotalMilliseconds);

            //Change the Timer's Due Time.

            Schedular.Change(dueTime, Timeout.Infinite);

        }

        catch(Exception ex)

        {

            WriteToFile("Simple Service Error on: {0} " + ex.Message + ex.StackTrace);

            //Stop the Windows Service.

            using (System.ServiceProcess.ServiceController serviceController = new System.ServiceProcess.ServiceController("SimpleService"))

            {

                serviceController.Stop();

            }

        }

    }

    private void SchedularCallback(object e)

    {

        this.WriteToFile("Simple Service Log: {0}");

        this.ScheduleService();

    }

    private void WriteToFile(string text)

    {

        string path = "C:\\ServiceLog.txt";

        using (StreamWriter writer = new StreamWriter(path, true))

        {

            writer.WriteLine(string.Format(text, DateTime.Now.ToString("dd/MM/yyyy hh:mm:ss tt")));

            writer.Close();

        }

    }

}

**VB.Net**

Public Class Service1

    Protected Overrides Sub OnStart(ByVal args() As String)

        Me.WriteToFile("Simple Service started at " + DateTime.Now.ToString("dd/MM/yyyy hh:mm:ss tt"))

        Me.ScheduleService()

    End Sub

    Protected Overrides Sub OnStop()

        Me.WriteToFile("Simple Service stopped at " + DateTime.Now.ToString("dd/MM/yyyy hh:mm:ss tt"))

        Me.Schedular.Dispose()

    End Sub

    Private Schedular As Timer

    Public Sub ScheduleService()

        Try

            Schedular = New Timer(New TimerCallback(AddressOf SchedularCallback))

            Dim mode As String = ConfigurationManager.AppSettings("Mode").ToUpper()

            Me.WriteToFile((Convert.ToString("Simple Service Mode: ") & mode) + " {0}")

            'Set the Default Time.

            Dim scheduledTime As DateTime = DateTime.MinValue

            If mode = "DAILY" Then

                'Get the Scheduled Time from AppSettings.

                scheduledTime = DateTime.Parse(System.Configuration.ConfigurationManager.AppSettings("ScheduledTime"))

                If DateTime.Now > scheduledTime Then

                    'If Scheduled Time is passed set Schedule for the next day.

                    scheduledTime = scheduledTime.AddDays(1)

                End If

            End If

            If mode.ToUpper() = "INTERVAL" Then

                'Get the Interval in Minutes from AppSettings.

                Dim intervalMinutes As Integer = Convert.ToInt32(ConfigurationManager.AppSettings("IntervalMinutes"))

                'Set the Scheduled Time by adding the Interval to Current Time.

                scheduledTime = DateTime.Now.AddMinutes(intervalMinutes)

                If DateTime.Now > scheduledTime Then

                    'If Scheduled Time is passed set Schedule for the next Interval.

                    scheduledTime = scheduledTime.AddMinutes(intervalMinutes)

                End If

            End If

            Dim timeSpan As TimeSpan = scheduledTime.Subtract(DateTime.Now)

            Dim schedule As String = String.Format("{0} day(s) {1} hour(s) {2} minute(s) {3} seconds(s)", timeSpan.Days, timeSpan.Hours, timeSpan.Minutes, timeSpan.Seconds)

            Me.WriteToFile((Convert.ToString("Simple Service scheduled to run after: ") & schedule) + " {0}")

            'Get the difference in Minutes between the Scheduled and Current Time.

            Dim dueTime As Integer = Convert.ToInt32(timeSpan.TotalMilliseconds)

            'Change the Timer's Due Time.

            Schedular.Change(dueTime, Timeout.Infinite)

        Catch ex As Exception

            WriteToFile("Simple Service Error on: {0} " + ex.Message + ex.StackTrace)

            'Stop the Windows Service.

            Using serviceController As New System.ServiceProcess.ServiceController("SimpleService")

                serviceController.[Stop]()

            End Using

        End Try

    End Sub

    Private Sub SchedularCallback(e As Object)

        Me.WriteToFile("Simple Service Log: " + DateTime.Now.ToString("dd/MM/yyyy hh:mm:ss tt"))

        Me.ScheduleService()

    End Sub

    Private Sub WriteToFile(text As String)

        Dim path As String = "C:\ServiceLog.txt"

        Using writer As New StreamWriter(path, True)

            writer.WriteLine(String.Format(text, DateTime.Now.ToString("dd/MM/yyyy hh:mm:ss tt")))

            writer.Close()

        End Using

    End Sub

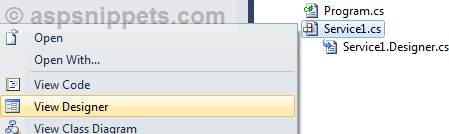
End Class

**Adding an Installer to the Windows Service**

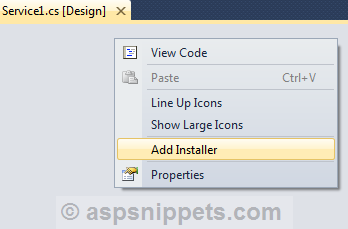
Once the Windows Service is ready to go we need to add the Installer class to our Windows Service as without it, the Windows Service will not install.

Following are the steps to add Installer class.

1. Right Click the Service1.cs class and click View Designer in the context menu.



2. Once the Design View is show, you need to right click and then select Add Installer in the context menu.



**Setting the Windows Service Name and StartType**

The above action will add an Installer class named ProjectInstaller. Now you need to open the ProjectInstaller.Designer class and look for InitializeComponent Method.

In this method we will modify the ServiceName of the Windows Service and also set its StartType to Automatic, so that along with the computer the Windows Service will start automatically.

**Note**: If you don’t set the StartType to Automatic, the default value is Manual and hence the Windows Service will not start automatically when the machine is started.

**C#**

private void InitializeComponent()

{

    this.serviceProcessInstaller1 = new System.ServiceProcess.ServiceProcessInstaller();

    this.serviceInstaller1 = new System.ServiceProcess.ServiceInstaller();

    //

    // serviceProcessInstaller1

    //

    this.serviceProcessInstaller1.Password = null;

    this.serviceProcessInstaller1.Username = null;

    //

    // serviceInstaller1

    //

    //Set the ServiceName of the Windows Service.

    this.serviceInstaller1.ServiceName = "SimpleService";

    //Set its StartType to Automatic.

    this.serviceInstaller1.StartType = System.ServiceProcess.ServiceStartMode.Automatic;

    //

    // ProjectInstaller

    //

    this.Installers.AddRange(new System.Configuration.Install.Installer[] {

    this.serviceProcessInstaller1,

    this.serviceInstaller1});

}

**VB.Net**

Private Sub InitializeComponent()

    Me.ServiceProcessInstaller1 = New System.ServiceProcess.ServiceProcessInstaller()

    Me.ServiceInstaller1 = New System.ServiceProcess.ServiceInstaller()

    '

    'ServiceProcessInstaller1

    '

    Me.ServiceProcessInstaller1.Password = Nothing

    Me.ServiceProcessInstaller1.Username = Nothing

    '

    'ServiceInstaller1

    '

    'Set the ServiceName of the Windows Service.

    Me.ServiceInstaller1.ServiceName = "SimpleService"

    'Set its StartType to Automatic.

    Me.ServiceInstaller1.StartType = System.ServiceProcess.ServiceStartMode.Automatic

    '

    'ProjectInstaller

    '

    Me.Installers.AddRange(New System.Configuration.Install.Installer() {Me.ServiceProcessInstaller1, Me.ServiceInstaller1})

End Sub

**Making the Windows Service Automatically start after Installation**

After the installation one has to start the Windows Service manually through the Services section of My Computer Management.

We can start the Windows Service automatically after installation by making use of the AfterInstall event handler which triggers immediately after Windows Service is installed.

You will need to open the ProjectInstaller class and override the AfterInstall event handler and add the code to start the Windows Service.

**C#**

[RunInstaller(true)]

public partial class ProjectInstaller : System.Configuration.Install.Installer

{

    public ProjectInstaller()

    {

        InitializeComponent();

    }

    protected override void OnAfterInstall(IDictionary savedState)

    {

        base.OnAfterInstall(savedState);

        //The following code starts the services after it is installed.

        using (System.ServiceProcess.ServiceController serviceController = new System.ServiceProcess.ServiceController(serviceInstaller1.ServiceName))

        {

            serviceController.Start();

        }

    }

}

**VB.Net**

Public Class ProjectInstaller

    Public Sub New()

        MyBase.New()

        'This call is required by the Component Designer.

        InitializeComponent()

        'Add initialization code after the call to InitializeComponent

    End Sub

    Protected Overrides Sub OnAfterInstall(savedState As IDictionary)

        MyBase.OnAfterInstall(savedState)

        'The following code starts the services after it is installed.

        Using serviceController As New System.ServiceProcess.ServiceController(ServiceInstaller1.ServiceName)

            serviceController.Start()

        End Using

    End Sub

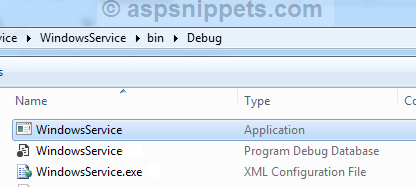
End Class

**Installing the Windows Service using InstallUtil.exe**

Once all the processes are complete, we can now build the Windows Service. Once the Windows Service is build you need to find the EXE file in the Debug folder of the Project.

**Note**: Once the Windows Service is ready for deployment, it is recommended to make use of the Release version of the EXE file instead of the Debug version.

To find the EXE, simply right click Project and select Open Folder in Windows Explorer. Now navigate to Bin => Debug folder and look for the EXE file with name same as that of the project.



Now copy and build the path in a Notepad (Text) file.

**Note**: I would recommend to build the command in a Notepad and save it somewhere so that you can use it multiple times.

InstallUtil Syntax

InstallUtil /i <Path of Windows Service EXE file>

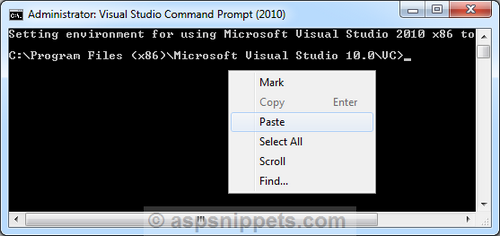
Example:

InstallUtil /i C:\Users\Mudassar\Projects\WindowsService\bin\Debug\WindowsService.exe

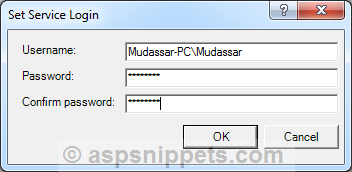
Now you need to open Programs => Microsoft Visual Studio 2010 => Visual Studio Tools => Visual Studio Command Prompt (2010).

**Note**: I am making use of Visual Studio 2010, and hence you need to use the appropriate version installed on your computer. And make sure you are logged in as Administrator. Without Administrator rights it would not allow you to install the Windows Service.

In the command prompt window, copy the InstallUtil command from Notepad and right click in the Command Prompt and click Paste and then press Enter key.

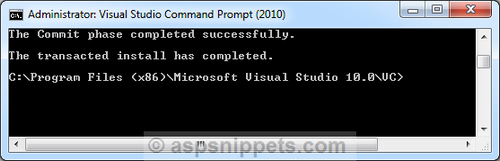


Now the Installer will ask for Logon permissions to run the Windows Service and hence you will need to add Windows Username and Password of user who has appropriate permission.

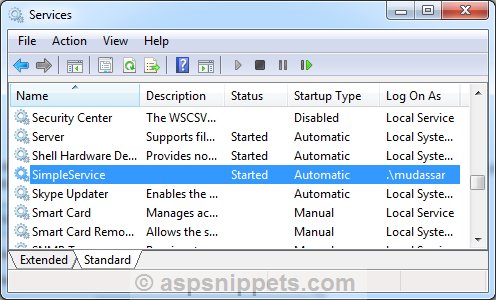


**Note**: Username must include Domain Name or the Computer name.

After successful installation you will see the following message.



You can find the Windows Service in the Services window. In order to open Services window in the Run Command type, services.msc and hit enter.



**Uninstalling the Windows Service using InstallUtil.exe**

The syntax for uninstalling a Windows Service is very similar to the installation syntax.

InstallUtil Syntax

InstallUtil /u <Path of Windows Service EXE file>

Example:

InstallUtil /u C:\Users\Mudassar\Projects\WindowsService\bin\Debug\WindowsService.exe

After successful uninstallation you will see the following message.

