**Introduction to Windows Services in .Net**

**Introduction**

This article explains a step-by-step process of developing and installing a Windows Service to do a scheduled job based on a time interval.

**What a Windows Service is**

Windows services are a special type of application that is configured to launch and run in the background. Services are designed to run continuously in the background and perform system tasks, like backing up your computer or running a server process that listens on a network port, etc.

**Why a Windows Service?**  
  
One of the most common requirements of some businesses is long-running scheduled jobs based on some time interval. For example: sending a newsletter everyday afternoon or send an email alert to the customer for every one hour.  
  
So building a Windows Service could be one of the reliable solutions to do the goal that can do the required job in the behind the scenes without interfering others users on the same computer

**Windows Service vs Job Scheduler**

Both can do most of the work you need, but both also have some advantages and disadvantages and based on your needs you should go with one or another.

The following are things you should take into concern

* **The frequency of the operation to be executed:** If you have to execute some operation not so often, it does not make sense to have piece of memory used most of the time. Let's say you send some emails once a week or daily. This is pretty much a scenario where you should go with scheduled task.

Otherwise, if you are running task very often, scheduled task is not so suitable as process may time more time than schedule pause and at some point

* **Communication:** If your scheduled operations needs to provide some data to other process, you should keep it as a service. Let's say you want to send push notifications to mobile devices. You are not sure about the frequency, which might be even one a day or once in few days, but you need to ensure that some other application can invoke this operation any time.

Scheduled task starts and ends, it does not reside in the memory. This scenario should be implemented as a service which resides in memory and can provide certain operation at any time.

* **The complexity of implementation:** Scheduled operation can be pretty much any executable on the Windows system. It can be simple console application, or it can have some UI which informs user that something is going on in a nice fancy dialog window, it can also be DOS batch (\*.bat) or command (\*.com) file.

Anyhow, you have more options how are you going to perform your operation which gives you certain range of flexibility to do it.

With services you do not have that freedom. Services are not so easy to debug (at least not easy as Windows Forms or console applications).

If you are not so good with code and you do not have some really complex operation, you should go with one of the options for scheduled task

* **Triggering mechanism:** While scheduled task can be only invoked after some time span expires, service have more options to initiate some processing.

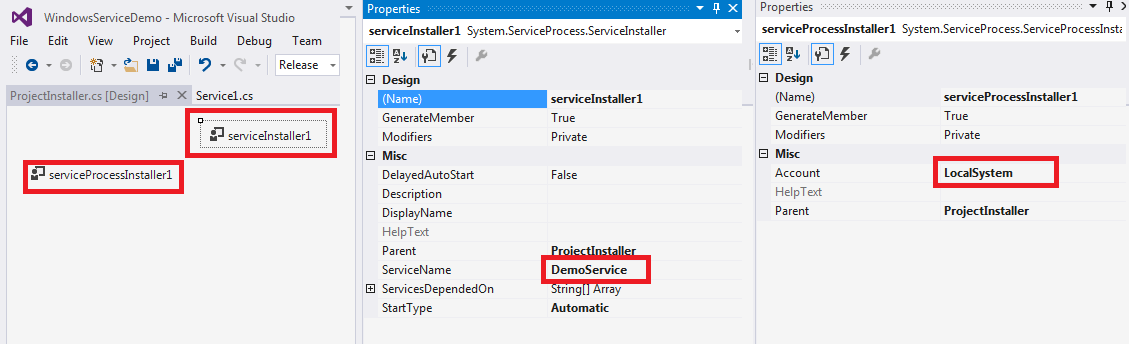
For example, if you need to perform some action when some file changes, you would have to use windows service which resides in memory and monitors specific file for a change.

**How to Create Windows Service**.

* To create windows services simply select "Windows Service” project template.
* A windows service basically has following three main events:
  + OnStart – To define what occurs when the service starts.
  + OnStop – To define what occurs when the service is stopped.
  + OnContinue - To define other actions for the service.

These methods automatically gets overridden from ServiceBase class when project is created.

* Specify Service name and User account by adding “ProjectInstaller” class to the project. Please refer screenshot below:



* To Install Windows Service Run following command from VS Developer Console by running it in Administrator mode

InstallUtil /i "PATH TO SERVICE EXECUTABLE"

* To Uninstall Windows Service Run following command from VS Developer Console by running it in Administrator mode

InstallUtil /u "PATH TO SERVICE EXECUTABLE"

Attached is the sample application to give you better idea how windows services works.

Reference:

<https://msdn.microsoft.com/en-us/library/zt39148a(v=vs.110).aspx>

<http://www.csharp-examples.net/create-windows-service/>

Windows Service,Service,C#,C Sharp