

<div>COLEXIO</div> <div>VIVAS</div> <div>S.L.</div>	RAMA:	Informática	CICLO:	DAM				
	MÓDULO	Programación de servicios y procesos					CURSO:	2º
	PROTOCOLO:	Apuntes clases	AVAL:		DATA:			
	UNIDAD		COMPETENCIA					

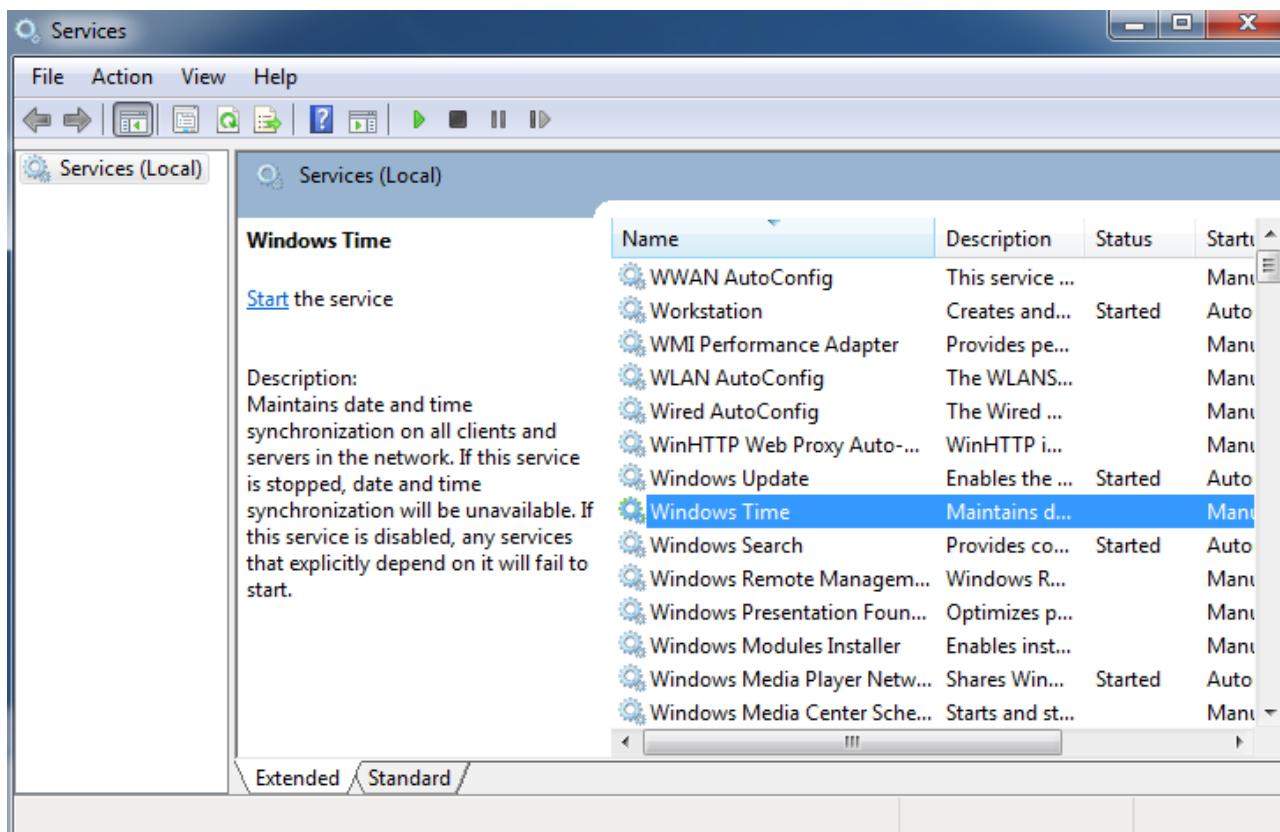
Creación de servicios en Windows

En este último tema de la asignatura, vamos a fundir todo lo aprendido anteriormente realizando las aplicaciones ya sin interface de usuario, trabajando en segundo plano y arrancando si fuera necesario con el sistema. El control se hará a partir del panel de control en la consola de servicios.

Consola de servicios de Windows

Los servicios del sistema operativo pueden ser controlados mediante una consola que se encuentra en el panel de control. Accediendo a Administrative Tools, y pulsando sobre Services accedemos a la misma.

También se puede ejecutar desde consola o Windows+R la aplicación **services.msc**.



COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	DAM
	MÓDULO	Programación de servicios y procesos		
	PROTOCOLO:	Apuntes clases	AVAL:	
	UNIDAD	COMPETENCIA	DATA:	

Aquí tenemos acceso a todos los servicios que están instalados en el sistema.

Un servicio no tiene porque estar en ejecución, de hecho tiene tres estados posibles:

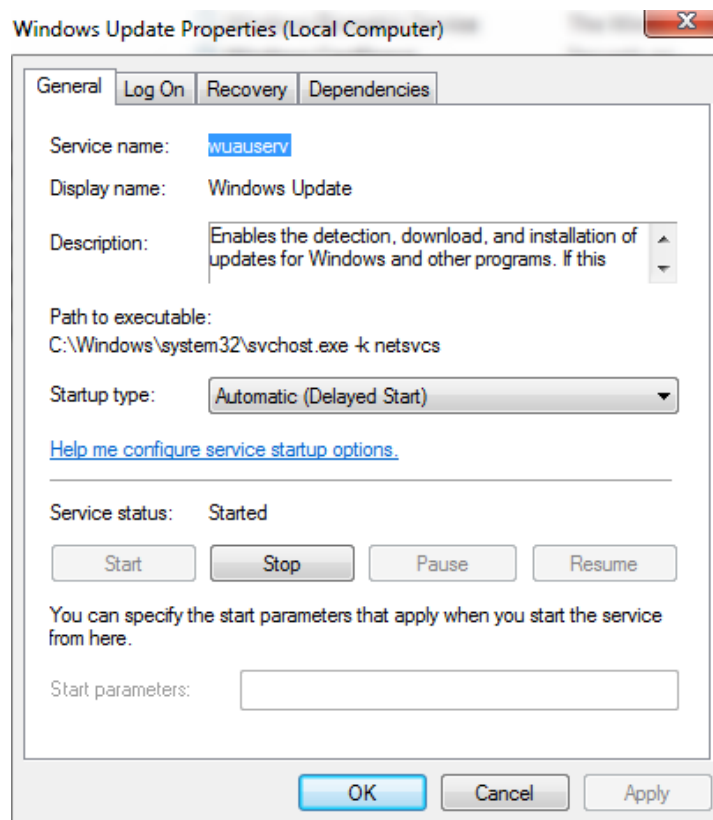
Stopped: No está en ejecución.

Running: está en ejecución.

Paused: Está en memoria pero temporalmente se ha parado el funcionamiento del mismo

En el panel también podemos ver a la derecha (si está la pestaña *Extended* habilitada) una descripción del mismo y enlaces para cambiar su estado.

También si hacemos doble clic sobre el servicio podemos acceder a su ventana particular de configuración:



COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Programación de servicios y procesos				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:		DATA:	
	UNIDAD	COMPETENCIA				

Disponemos de varias pestañas:

General: Descripción y configuración general sobre el servicio (programa, ruta, forma en la que arranca). También aparecen los botones para cambiarlo de estado.

Log On (Iniciar Sesión): Indica con qué permisos debe arrancar un servicio.

Recovery (Recuperación): En caso de fallo del servicio, podemos tomar diversas acciones como volver a lanzarlo, reiniciar el ordenador, etc...

Dependencies (Dependencias) Indica de qué servicios depende y qué otros servicios dependen de él.

Es necesario conocer todas estas propiedades (o al menos saber que existen), ya que cuando programemos un servicio tendremos forma de configurarlas en nuestro código.

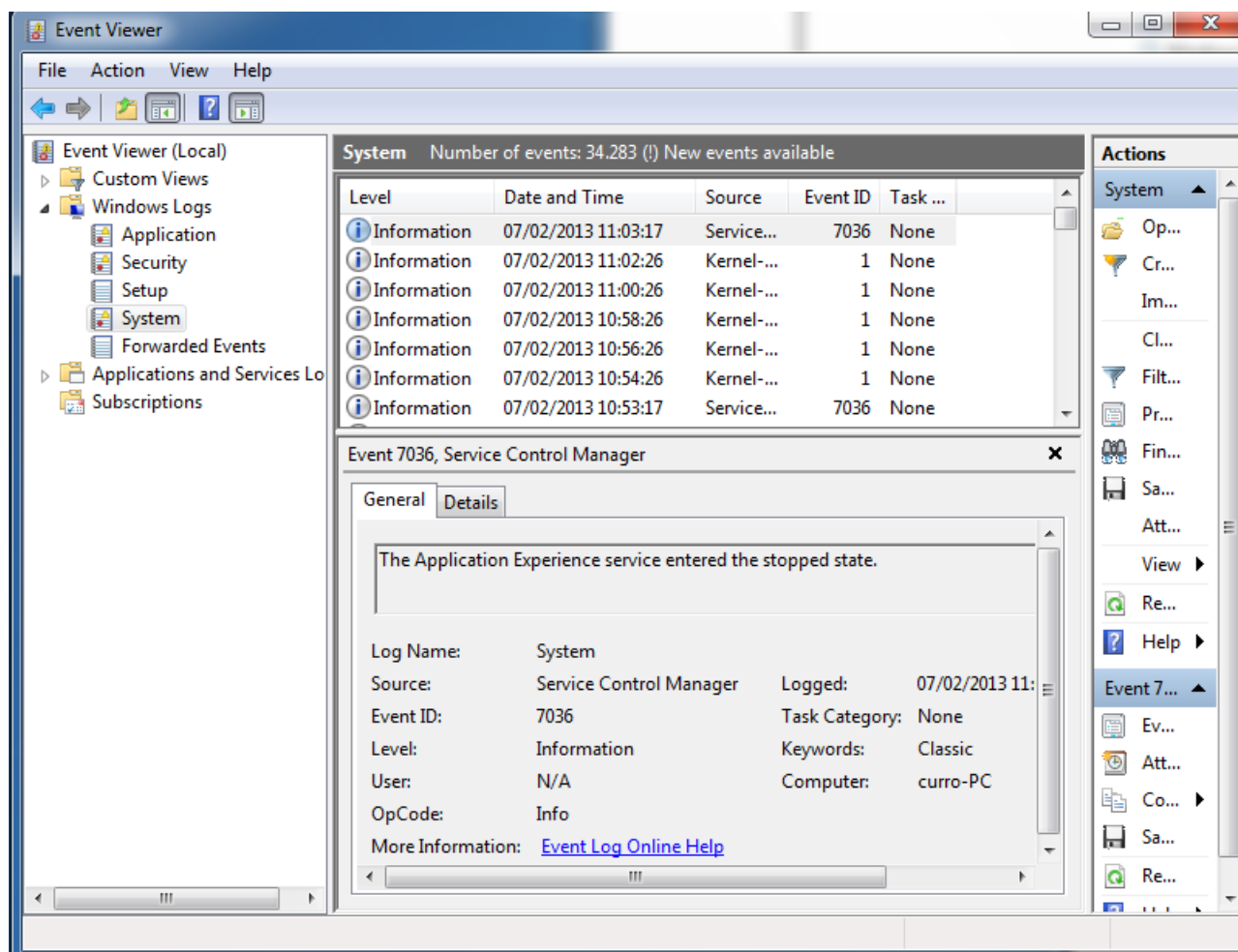
También resulta importante conocer el visor de sucesos o eventos (**Event Viewer**) que es otra consola que nos permite ver una serie de ficheros de log del propio Windows. Entre otros eventos que ahí aparecen están el inicio, pausa, re arranque o parada de un servicio (siempre que esté programado para ello).

También se puede ejecutar como **eventvwr.msc**.


Dicho visor se encuentra también en Administrative Tools, y dentro del apartado **Windows Logs** → **System** o en **Windows Logs** → **Application** aparecen generalmente los mensajes de los servicios.

Experimenta un poco con ella para ir recordando y entendiendo la información que muestra y cómo la muestra.

<div>COLEXIO</div> <div>VIVAS</div> <div>S.L.</div>	RAMA:	Informática	CICLO:	DAM				
	MÓDULO	Programación de servicios y procesos					CURSO:	2º
	PROTOCOLO:	Apuntes clases	AVAL:		DATA:			
	UNIDAD		COMPETENCIA					



Una vez que tenemos claro el funcionamiento de estas dos utilidades relacionadas con los servicios podremos pasar a la programación de los mismos.

	RAMA:	Informática	CICLO:	DAM				
	MÓDULO	Programación de servicios y procesos					CURSO:	2º
	PROTOCOLO:	Apuntes clases	AVAL:		DATA:			
	UNIDAD		COMPETENCIA					

Programación básica de servicios

Un servicio es un programa ejecutable (.exe) que además tiene una serie de características que lo hacen especial.

Para empezar no se ejecuta con un simple Doble-clic sobre el archivo, si no que debe estar registrado (instalado) en el sistema operativo de una forma concreta para poder ser gestionado por el mismo SO y además desde la consola de servicios vista anteriormente (Services).

También debe tener programado qué realizar ante eventos de parada (Stop), arranque (Start), pausa (Pause) y continuación (Resume) a través de los botones de la consola de servicios.

Por todo esto para la realización de un servicio con estas características hay que cumplir unas reglas que no son más que adaptarse a una determinada estructura heredando ciertas clases y sobrescribiendo algunas de sus funciones.

Luego, tras compilar, pasaremos a realizar la instalación del servicio ejecutable en el sistema mediante la aplicación **InstallUtil**.

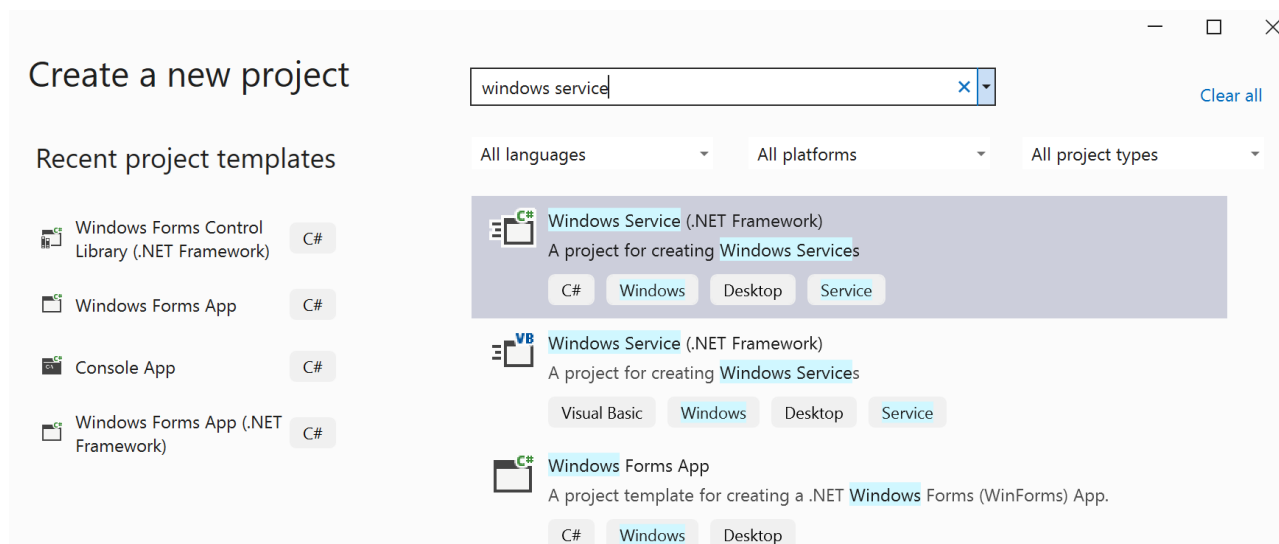
Para entender el proceso, vamos a crear un servicio sencillo mediante un Timer que almacena información cada cierto tiempo en el Event Viewer.

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Programación de servicios y procesos				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:		DATA:	
	UNIDAD	COMPETENCIA				

Ejemplo guiado: instalando un servicio en el panel de servicios

a) Preparación del servicio: Clase ServiceBase

Abre Visual Studio y crea un nuevo proyecto. Selecciona la plantilla **Windows Service (.Net Framework)** de c# (Servicio de Windows (.Net Framework)).



Este paso nos dispone una clase que hereda de ServiceBase.

Por un lado aparece un diseñador para añadir componentes como si fuera una aplicación de formulario. Realmente el objetivo es añadir ciertos componentes que puedan tener cierta utilidad para el servicio más que el hecho de tener una interfaz de usuario. Podría usarse, por ejemplo, para añadir un objeto del tipo EventLog, que nos permite escribir mensajes en el visor de eventos.

Esta clase base dispone de métodos OnStart, OnStop (estos aparecen por defecto) y otros que permiten el manejo del servicio y que serán sobrescritos. Además dispone de una serie de propiedades que nos da una configuración inicial para dicho servicio.

Nos aparece cierto código que está aquí ampliado y comentado:

<div>COLEXIO</div> <div>VIVAS</div> <div>S.L.</div>	RAMA:	Informática	CICLO:	DAM					
	MÓDULO	Programación de servicios y procesos						CURSO:	2º
	PROTOCOLO:	Apuntes clases	AVAL:		DATA:				
	UNIDAD		COMPETENCIA						

```

using System;
using System.ServiceProcess;

namespace MyService
{
    public partial class Servicio1:ServiceBase
    {
        // Constructor donde se inicializan algunas propiedades de interés
        public Servicio1()
        {
            InitializeComponent();
        }

        //Se ejecuta cuando inicia el servicio
        protected override void OnStart(string[] args)
        {
        }


        //Se ejecuta cuando para el servicio
        protected override void OnStop()
        {
        }

        //Se ejecuta cuando pausa el servicio
        protected override void OnPause()
        {
        }

        //Se ejecuta cuando continua tras la pausa el servicio
        protected override void OnContinue()
        {
        }
    }
}

```

Properties	
Servicio1 System.ServiceProcess.ServiceBase	
(Name)	Servicio1
AutoLog	True
CanHandlePowerEvent	False
CanHandleSessionChange	False
CanPauseAndContinue	False
CanShutdown	False
CanStop	True
ExitCode	0
Language	(Default)
Localizable	False
ServiceName	Servicio1

	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Programación de servicios y procesos				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:		DATA:	
	UNIDAD	COMPETENCIA				

Bien, vamos a aplicar esto creando un nuevo servicio. En vez de lanzar como hilo un servidor nuestro, usaremos un Timer que guarde cada cierto tiempo información en el Log, pero la funcionalidad es la misma

1. Si aún no lo has hecho crea un nuevo proyecto denominado MyService usando la plantilla Servicio de Windows.

Esto crea un proyecto con una nueva clase: Service1 que cumple la plantilla de herencia de ServiceBase que se vio previamente.

2. Cambiamos el nombre en la ventana de propiedades en propiedad ServiceName (este será el nombre con que lo identifique el sistema) y en (Name) que será el nombre de la clase. En ambos casos lo denominamos SimpleService.

Esto no tiene por que coincidir con el nombre que se pondrá luego en el panel de servicios.

Puedes cambiar también el nombre en el explorador de soluciones a MyService.cs.

Tendremos el siguiente código (obviando los using):

```
namespace MyService
{
    public partial class SimpleService : ServiceBase
    {
        public SimpleService()
        {
            InitializeComponent();
        }

        protected override void OnStart(string[] args)
        {
        }

        protected override void OnStop()
        {
        }
    }
}
```

3. Crea una función denominada writeEvent (ponla antes del OnStart) que servirá

<div>COLEXIO</div> <div>VIVAS</div> <div>S.L.</div>	RAMA:	Informática	CICLO:	DAM				
	MÓDULO	Programación de servicios y procesos					CURSO:	2º
	PROTOCOLO:	Apuntes clases	AVAL:		DATA:			
	UNIDAD		COMPETENCIA					

para escribir mensajes en el visor de eventos. Podría hacerse como se comentó con un componente eventLog, pero como vamos a darle un uso simple usaremos directamente las funciones estáticas de la clase EventLog.

La fuente (source) es el nombre de la aplicación que genera el log, y el logDestino es, dentro de la jerarquía del visor de eventos, donde queremos que se coloquen nuestros logs. Lo habitual es hacerlo en System o Application.

```
public void writeEvent(string mensaje)
{
    string nombre = "SimpleService";
    string logDestino = "Application";

    if (!EventLog.SourceExists(nombre))
    {
        EventLog.CreateEventSource(nombre, logDestino);
    }

    EventLog.WriteEntry(nombre, mensaje);
}
```

4. A continuación vamos a realizar cierta programación en el método OnStart.


Es muy importante aclarar que aunque un servicio sea un programa que se ejecuta a largo plazo, el método OnStart debe realizar pocas tareas y finalizar en cuanto las acabe.

Así el uso habitual del OnStart es la inicialización y lanzamiento de lo que sería el hilo principal del servicio.

En este caso sencillo, el hilo se gestiona a través del Timer y sería algo así:

```
protected override void OnStart(string[] args)
{
    writeEvent("Running OnStart");
    System.Timers.Timer timer = new System.Timers.Timer();
    timer.Interval = 10000; // cada 10 segundos
    timer.Elapsed += new System.Timers.ElapsedEventHandler(this.OnTimer);
    timer.Start();
}
```

En un caso real, en vez de un Timer crearás y lanzarás un hilo con la función de inicio de tu servicio. **Solo lanzar el hilo, nada más, y que OnStart finalice.**

	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Programación de servicios y procesos				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:		DATA:	
	UNIDAD	COMPETENCIA				

5. Añade el método TimerTick que se ejecutará cada 10 segundos simplemente mostraremos en el visor de eventos un mensaje con un contador:

```
private int t = 0;
public void TimerTick(object sender, System.Timers.ElapsedEventArgs args)
{
    writeEvent(string.Format($"SimpleService running about {t} seconds"));
    t += 10;
}
```

6. Sobreescribe otros método para ver parte del ciclo de vida del servicio como OnStop o OnPause y OnContinue (Para que esto último se permita en el panel, debes poner cierta propiedad a true en MyService, búscala):

```
protected override void OnPause()
{
    writeEvent("Servicio en Pausa");
}

protected override void OnContinue()
{
    writeEvent("Continuando servicio");
}

protected override void OnStop()
{
    writeEvent("Deteniendo servicio");
    t = 0;
}
```

Por supuesto para gestionar estos métodos correctamente en un caso real, y que se comuniquen adecuadamente con el Servicio, **hay que establecer booleanas, usar lock, Wait, Pulse, IsBackground** u otros elementos vistos en hilos.

Con esto ya tenemos creado el servicio, el problema es que no se puede ejecutar directamente si no que hay que instalarlo como servicio válido de Windows. Esto es lo que haremos a continuación.

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Programación de servicios y procesos				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:		DATA:	
	UNIDAD	COMPETENCIA				

b) Instalación del servicio 1ª parte: Proyecto de instalación

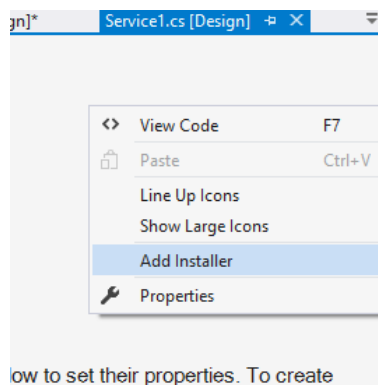
Por ser ejecutable tendrá que tener un Main. Como en el caso de formularios es corto y lo único que se limita es a lanzar el servicio.


```
namespace MyService
{
    internal static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        static void Main()
        {
            ServiceBase[] ServicesToRun;
            ServicesToRun = new ServiceBase[]
            {
                new SimpleService()
            };
            ServiceBase.Run(ServicesToRun);
        }
    }
}
```

Como el servicio tiene que estar registrado en el sistema operativo necesitamos una clase que se encargue de realizar esta instalación y que indique algunas características de funcionamiento del servicio como la cuenta de usuario en la que se instala, el modo de arranque, la descripción y otros.

Los pasos a seguir son los siguientes:

1. Primero es necesario añadir un instalador. **Sobre la vista de Diseño** del servicio abrimos el menú contextual y seleccionamos Add Installer.



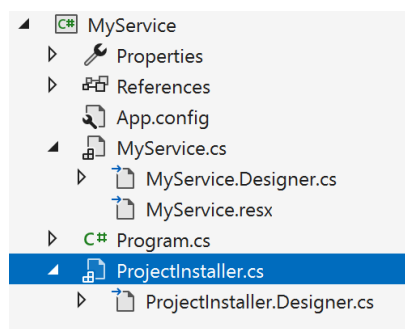
	RAMA:	Informática	CICLO:	DAM				
	MÓDULO	Programación de servicios y procesos					CURSO:	2º
	PROTOCOLO:	Apuntes clases	AVAL:		DATA:			
	UNIDAD		COMPETENCIA					

Esto añade al proyecto la clase ProjectInstaller que sirve para facilitar la instalación de servicios mediante dos objetos: Uno tipo ServiceInstaller y otro tipo ServiceProcessIntaller

Nota: Si quisieras realizar un instalador para una aplicación cualquiera se utiliza una solución del tipo Setup Wizard. Puedes ver más información aquí:

<https://stackoverflow.com/questions/6090913/make-an-installation-program-for-c-sharp-applications-and-include-net-framework>

Debes tener el Explorador de soluciones de la siguiente forma:



Además activa el atributo RunInstaller es necesario para indicar a la aplicación InstallUtil cuál es la clase que contiene la información de instalación en el Proyecto. Lo puedes ver al acceder al código de ProjectIntaller:

```
[RunInstaller(true)]
```

2. El objeto de la clase **ServiceProcessInstaller** es necesario para indicarles ciertas propiedades del futuro proceso al instalador. Es decir, cualquier aplicación, sea servicio o aplicación normal, tiene ciertas características como la cuenta con la que se ejecuta. Esto se establece en este objeto.

Selecciona este objeto y en propiedades establece la **propiedad Account** para que el servicio sea considerado como servicio del sistema local (**LocalSystem**) y no necesite iniciarse con cuenta de usuario.

Si no le das estos privilegios, no puede acceder, entre otros, al visor de eventos.

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Programación de servicios y procesos				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:		DATA:	
	UNIDAD	COMPETENCIA				

Puedes leer más sobre esta propiedad en:

<https://docs.microsoft.com/es-es/dotnet/api/system.serviceprocess.serviceaccount?view=netframework-4.7.2>

3. El objeto de la clase **ServiceInstaller** indica propiedades más particulares del servicio como el nombre a mostrar en el panel de servicios, la descripción, etc.

Selecciona este último y ve a propiedades.

Establece ahí **ServiceName** a **SimpleService** (si no lo ha hecho ya automáticamente) que será el nombre que aparecerá en el panel de servicios.

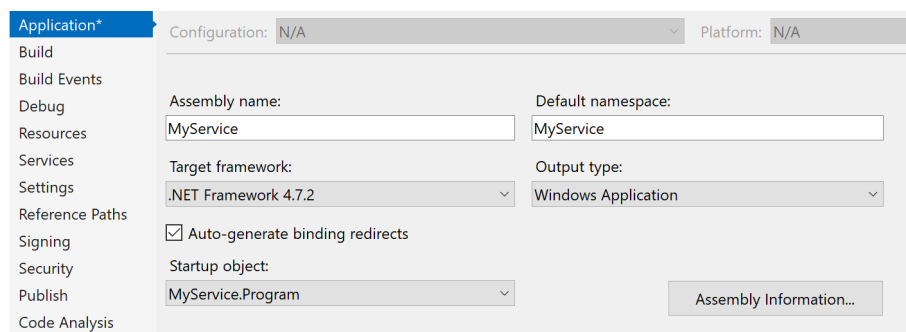
Otras propiedades que puedes configurar:

Description (la descripción que aparecerá en el panel de servicios): Por ejemplo "Simple Test Service".

DisplayName: (nombre con el que aparece el servicio en el panel de servicios, puede ser distinto al Service Name): "My Simple Service"

StartType (indica quién inicia el servicio, al ponerlo en automático se inicia con el SO): Déjalo en Manual para inicializarlo mediante el panel de servicios.

4. En el explorador de soluciones con el botón derecho sobre el proyecto selecciona propiedades. En el apartado aplicación, en la lista **Startup Project** (Objeto de Inicio) selecciona **MyService.Program**.



5. Finalmente compila la solución (No ejecutes).

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Programación de servicios y procesos				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:		DATA:	
	UNIDAD	COMPETENCIA				

c) Instalación del servicio 2ª parte: Comando InstallUtil

En el apartado anterior creamos un ejecutable que no se puede ejecutar directamente, si no que hay que instalarlo como servicio.

Usaremos el programa **InstallUtil** que normalmente se encuentra en

`C:\Windows\Microsoft.NET\Framework\v4.0.30319`

Dependiendo del VS instalado o la versión del Framework usada los directorio pueden variar (echa un ojo y verás que hay otros similares).

Tenemos que abrir una consola en modo administrador. Se puede usar el CMD habitual o la **consola para desarrolladores que incorpora Visual Studio**. La ventaja de esta última es que ya incorpora en el PATH diversas utilidades como InstallUtil y no es necesario ir al directorio mencionado anteriormente.

Para abrir esta última en el menú de inicio busca **Developer command prompt for VS 2022** (Simbolo de Sistema para desarrolladores VS 2022) dentro de la carpeta del VisualStudio 2022. **Recuerda abrirla como administrador.**

Ve al directorio donde tienes el compilado (MyService.exe) de tu proyecto y ejecuta la siguiente línea:

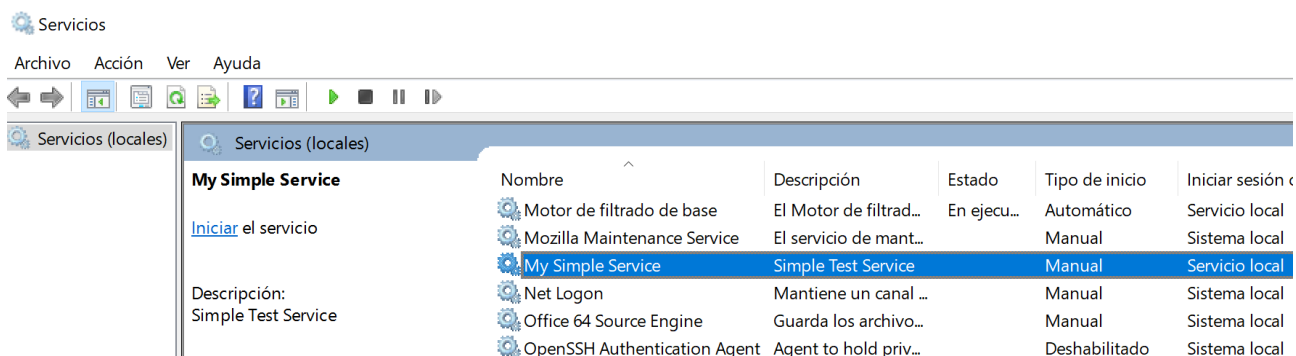
`InstallUtil.exe NuevoServicio.exe`

Salen varios mensajes entre ellos, si no hubo problema, te indicará que las fases de instalación y confirmación finalizaron correctamente y que la instalación ha finalizado.

Vete al panel de control de servicios , actualízalo (F5) y busca el nuevo servidor haciendo doble clic sobre el mismo.

Deberías tener algo así:

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Programación de servicios y procesos				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:		DATA:	
	UNIDAD	COMPETENCIA				



En las propiedades del servicio se puede ver que este nuevo servicio cumple los parámetros que hemos programado. Se puede iniciar, parar pausar y hacer que continúe (Resume).

Si te da **acceso denegado** es que no has configurado bien la propiedad Account.

Si te da un **SecurityException** es posible que no estés ejecutando como administrador la consola.

Prueba que todo funcione correctamente.

Vete al visor de eventos para ver los mensajes que se van dejando. Comprobarás que además de los que tu has programado aparecen otros de manera automática. Busca la propiedad adecuada para desactivar dichos mensajes automáticos.

<div>COLEXIO</div> <div>VIVAS</div> <div>S.L.</div>	RAMA:	Informática	CICLO:	DAM				
	MÓDULO	Programación de servicios y procesos					CURSO:	2º
	PROTOCOLO:	Apuntes clases	AVAL:		DATA:			
	UNIDAD		COMPETENCIA					

Visor de eventos

Archivo Acción Ver Ayuda



- Visor de eventos (local)
- > Vistas personalizadas
- ▼ Registros de Windows
 - Aplicación
 - Seguridad
 - Instalación
 - Sistema
 - Eventos reenviados
- > Registros de aplicaciones y servicios
- Suscripciones

Aplicación
Número de eventos: 29.285 (!) Nuevos eventos disponibles

Nivel	Fecha y hora	Origen
Información	14/02/2023 11:43:26	SimpleService
Información	14/02/2023 11:43:25	SimpleService
Información	14/02/2023 11:43:22	SimpleService

Evento 0, SimpleService

General Detalles

SimpleService running about 0 seconds

Nombre de registro: Aplicación

Origen: SimpleService Registrado: 14/02/2023 11:43:25

Id. del: 0 Categoría de tarea: Ninguno

Nivel: Información Palabras clave: Clásico

Usuario: No disponible Equipo: win81parallels

Código de operación: Información

Más información: [Ayuda Registro de eventos](#)

Si el servicio falla tenemos que depurarlo, recompilarlo y reinstalarlo. Pero el sistema no nos deja si dicho servicio ya está instalado. **Hay que desinstalar previamente** el que no vale. Para ello se usa también InstallUtil pero con el parámetro /u (uninstall).


Sería así:

`InstallUtil.exe /u NuevoServicio.exe`

Si da algún problema cierra la consola de Servicios y el Visor de Eventos.

Más información en:

[https://msdn.microsoft.com/es-es/library/zt39148a\(v=vs.110\).aspx](https://msdn.microsoft.com/es-es/library/zt39148a(v=vs.110).aspx)

	RAMA:	Informática	CICLO:	DAM				
	MÓDULO	Programación de servicios y procesos					CURSO:	2º
	PROTOCOLO:	Apuntes clases	AVAL:		DATA:			
	UNIDAD		COMPETENCIA					


Línea futura de trabajo

Quedan varias cosas pendientes que no veremos, la más importante en principio es quizá la de hacer el servicio configurable, ya que no se puede hacer interfaz de usuario. Lo que se suele hacer (recomendación de Microsoft) es usar el WCF (Windows Communication Foundation), pero no lo vamos a ver.

Siempre puedes tener los elementos de configuración en un archivo y mediante un frontend (o un editor de texto) cambies dichos parámetros (esta fórmula de trabajo es muy habitual en Linux).

Una vez modificado el archivo es cuestión de parar y reiniciar el servicio para que cargue los nuevos valores de configuración.

Por supuesto se debe programar el servicio para que lea dicho archivo nada más arrancar.


	RAMA:	Informática	CICLO:	DAM				
	MÓDULO	Programación de servicios y procesos					CURSO:	2º
	PROTOCOLO:	Apuntes clases	AVAL:		DATA:			
	UNIDAD		COMPETENCIA					

Ejercicio

Toma algún servidor hecho en el tema de networking (servidor de fecha y hora, chatroom,...) y adáptalo (el cliente, si lo hay, no hay que tocarlo) para que funcione como un servicio de Windows.

Ten en cuenta las siguientes características:

- No debe poderse pausar el servicio.
- Debe leer el puerto de escucha de un archivo de configuración que se encontrará en %PROGRAMDATA%. Si no existe el archivo coge un puerto por defecto.
- En el visor de eventos se informará del puerto de escucha y del error al leer archivo si sucede. Si al final tanto el puerto leído del archivo como el puerto por defecto estuvieran ocupados, se informará en el visor de eventos y finalizará el servicio.
- Para la validación debes mostrarlo al profesor de la materia de forma presencial o enviar un video donde se vea claramente los siguientes elementos:
 - Servicio instalado en el panel con la descripción clara.
 - Se arrancará y parará el servicio 2 veces.
 - Se mostrará el puerto de escucha en el visor de eventos
 - Tras esa segunda vez se hará una prueba de conexión al mismo desde telnet y con algún envío de comandos para ver que funciona correctamente.

	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Programación de servicios y procesos				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:		DATA:	
	UNIDAD	COMPETENCIA				

Bibliografía

<http://noshonet.blogspot.com.es/2012/03/creacion-de-servicios-de-windows-c-net.html>

<http://stackoverflow.com/questions/569606/how-can-i-make-service-apps-with-visual-c-sharp-express>

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

<http://msdn.microsoft.com/en-us/library/>

<div>COLEXIO</div> <div>VIVAS S.L.</div>	RAMA:	Informática	CICLO:	DAM				
	MÓDULO	Programación de servicios y procesos					CURSO:	2º
	PROTOCOLO:	Apuntes clases	AVAL:		DATA:			
	UNIDAD		COMPETENCIA					

Apéndice: Escritura de mensajes en el Visor de Eventos (Event Viewer)

En la práctica guiada crearemos un servicio similar a uno visto anteriormente: el servidor de eco, pero le añadiremos la posibilidad de escribir información en el Visor de eventos. Será a través de un método como el siguiente (Es necesario el espacio de nombre *System.Diagnostics*):

```
public void escribeEvento(string mensaje)
{
    string nombre = "Servidor de Echo"; //Nombre del servicio
    string logDestino = "Application"; // Donde aparece el evento (debe
existir)
    //Se puede probar otro nombre y ver cómo da error
    //En el propio visor de eventos

    if (!EventLog.SourceExists(nombre))
    {
        EventLog.CreateEventSource(nombre, logDestino);
    }

    EventLog.WriteEntry(nombre, mensaje);
    //Existe más sobrecargas donde se indica Icono, ID, etc.
}
```

Es relativamente fácil de entender: Se comprueba que exista la aplicación en el Gestor de Eventos, si no existe se crea (esto se hará solo la primera vez o mejor incluso durante la instalación). Tras ello se usa el método *WriteEntry* para dejar el mensaje.

Tiene diversos parámetros más pero no vamos a entrar en ellos. Tienen normalmente relación con la información del evento que aparece en la parte inferior del visor de eventos cuando se soluciona uno.

Lo que es importante es tener privilegios suficientes para poder ejecutar los comandos *SourceExists* y *CreateEventSource*, por lo que si no interesa que la aplicación o el servicio tenga estos privilegios, se puede crear el Event Source en el instalador, ya que luego, una vez creada la fuente ya no da ningún problema.

Ahora para hacer pruebas se recomienda añadir al proyecto un manifiesto y cambiar los privilegios de *"asInvoker"* a *"requireAdministrator"*.

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Programación de servicios y procesos				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:		DATA:	
	UNIDAD	COMPETENCIA				

Más información en:

<https://support.microsoft.com/es-es/help/307024/how-to-write-to-an-event-log-by-using-visual-c>

[https://msdn.microsoft.com/es-es/library/system.diagnostics.eventlog.writeentry\(v=vs.110\).aspx](https://msdn.microsoft.com/es-es/library/system.diagnostics.eventlog.writeentry(v=vs.110).aspx)