Resumen Nivel 2, Leccion 3

Introducción a Desarrollo de Software

Daniel Germán - 1128127

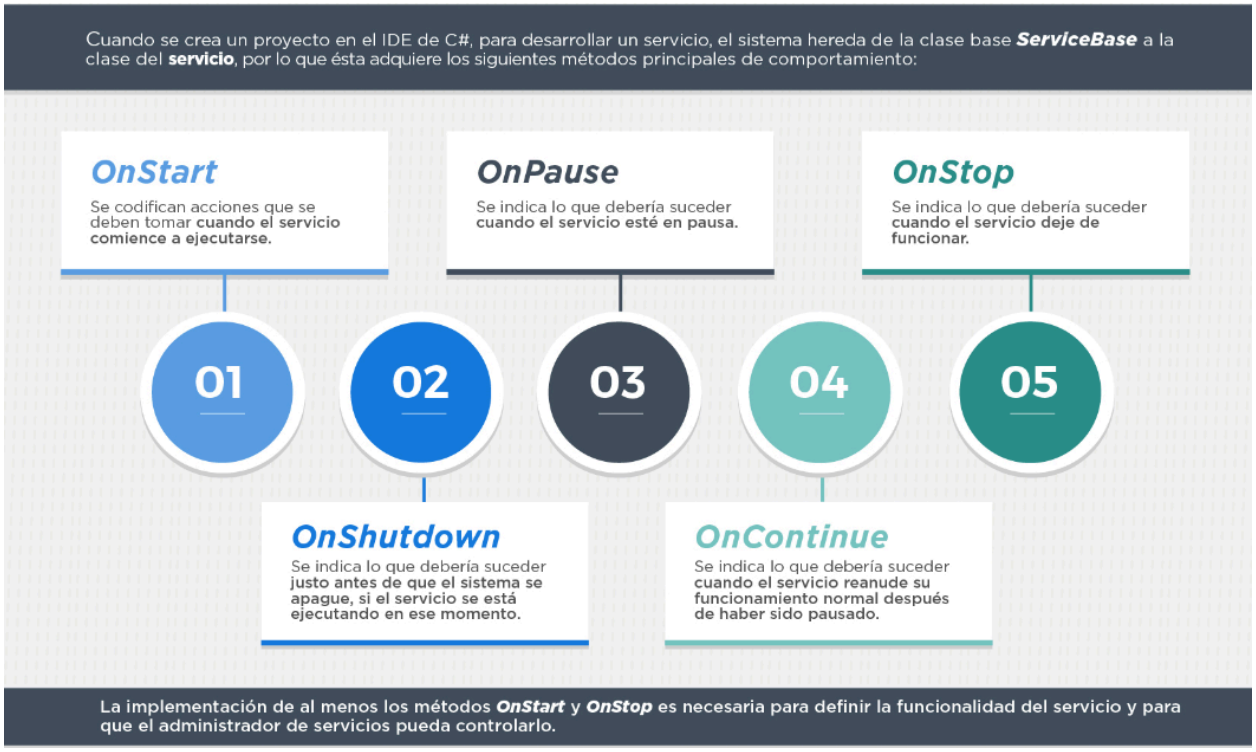
15 de diciembre, 2024

Introducción a servicios locales

Un servicio local es un programa interno o de una aplicación que se ejecuta en segundo plano, por lo que no tiene una interfaz y no se puede ver por el usuario de manera directa. Siempre están ejecutados. Los Update Services son un ejemplo de estos que corren con frecuencia, ya que no tienen una interfaz pero hacen checks para ver si hay actualizaciones disponibles para el dispositivo, que luego se muestran al usuario. El servicio no tiene fin a menos que el usuario lo desactive mientras que un programa finaliza al llegar al fin de su método Main. Un administrador de servicio permite ver, manejar y configurar los servicios del dispositivo.

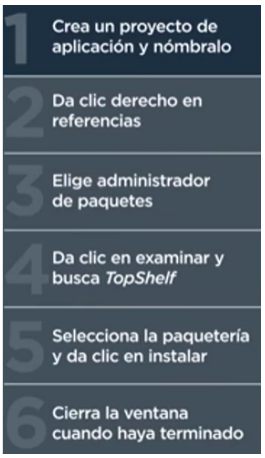
Hay cinco maneras de iniciar un servicio:

1. Automático, lo inicia el sistema por sí mismo
2. Inicio retrasado, inicia despues de que el OS esté completamente iniciado
3. Manual, lo inicia el usuario vía un software
4. Desencadenar, solo inicia si hay espacio y recursos para iniciar el servicio
5. Deshabilitado, no se puede iniciar ya que fue deshabilitado por el usuario

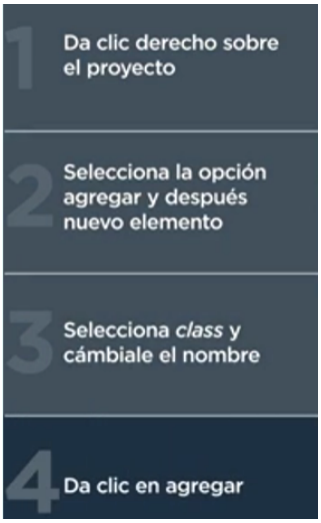


Proyecto de servicio local

En el IDE de C#, programar un servicio local no utiliza el depurador de código. Para ello, se tiene que codificar el servicio, instalarlo, y verificar su funcionamiento. En caso de que no funcione, hay que desinstalarlo, corregir el código para instalarlo y probarlo de nuevo. Para facilitar este proceso, se puede usar un administrador de paquetes:



TopShelf es una paquetería que ayuda a verificar la funcionalidad de servicios locales. El servicio debe estar desagregado de la clase que contiene al método Main. Para hacer esto, se debe crear una clase:

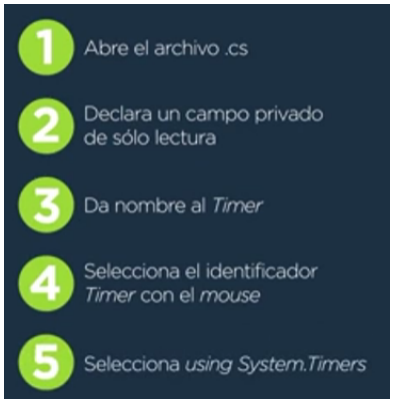


Hasta ahora, el framework para crear un servicio está hecho, por lo que solo falta programarlo y probarlo.

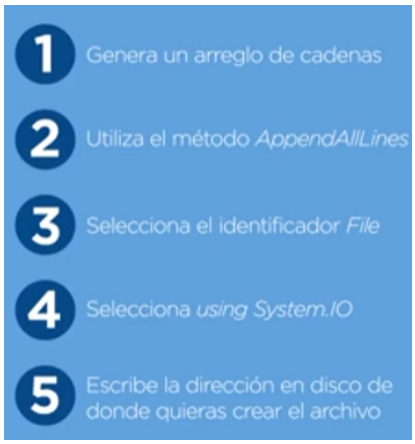
Creación de un servicio local

Para crear un servicio local, se deben definir dentro de la clase un método constructor, un método de inicio, un método de parada, un timer, y un manejador de eventos. Un servicio siempre debe

marcar el tiempo en el que debe ser ejecutado. Para definir el timer:



El funcionamiento del servicio va dentro del constructor.

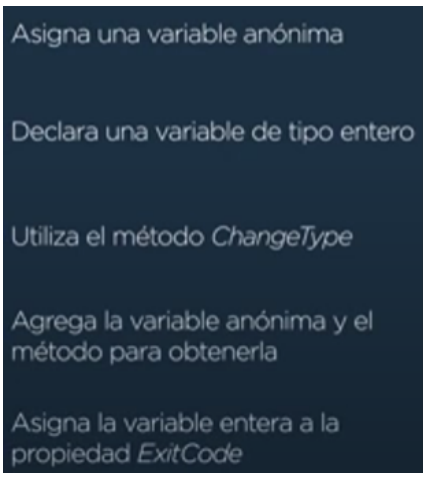


A continuación, se declara cuando debe iniciar y finalizar el servicio. Esto se logra definiendo dos public voids que inician y detienen el timer. Luego, todo se ve dividido en las partes Front, Back, y Service.

Front es lo que aparece en el administrador de servicios, Back se refiere a los componentes que se encuentran como bases del servicio como el constructor, el inicio y el detener, mientras que el servicio instancia el objeto de la clase del servicio. Para habilitar el servicio, se debe acceder al programa.cs y escribir el método Main.

Luego, se mandan al administrador de paquetes las variables necesarias para ejecutar el servicio. Estas variables son métodos:

Al definir estos valores, se puede iniciar la definición del back. Finalmente, para lograr un cierre del código, se utiliza:



Para instalar un servicio local usando Topshelf, se abre una terminal como administrador, se navega a la carpeta del ejecutable y ejecutar <nombre_del_ejecutable>.exe install. Para desinstalarlo, usar <nombre_del_ejecutable>.exe uninstall.

Actividad

Program.cs

```
using System;
using Topshelf;

namespace DeleteFileService
{
    class Program
    {
        static void Main(string[] args)
        {
            HostFactory.Run(config =>
            {
                config.Service<FileDeleteService>(service =>
                {
                    service.ConstructUsing(() => new FileDeleteService());
                    service.WhenStarted(s => s.Start());
                    service.WhenStopped(s => s.Stop());
                });

                config.RunAsLocalSystem();

                config.SetServiceName("FileDeleteService");
                config.SetDisplayName("File Delete Service");
                config.SetDescription("Servicio que elimina un archivo
cada hora.");
            });
        }
    }
}
```

Service1.cs

```
using System;
using System.IO;
using System.Timers;

namespace DeleteFileService
{
    public class FileDeleteService
    {
        private readonly Timer _timer;

        public FileDeleteService()
        {
            // Configura el temporizador para ejecutarse cada hora
(3600000 ms)
            _timer = new Timer(3600000);
            _timer.AutoReset = true;
            _timer.Elapsed += Timer_Elapsed;
        }

        private void Timer_Elapsed(object sender, ElapsedEventArgs e)
        {
            string filePath = @"C:\Users\zerol\Desktop\FileUpdate.txt";

            try
            {
                if (File.Exists(filePath))
                {
                    File.Delete(filePath);
                    Console.WriteLine($"{DateTime.Now}: Archivo
eliminado.");
                }
                else
                {
                    Console.WriteLine($"{DateTime.Now}: El archivo no
existe.");
                }
            }
            catch (Exception ex)
```

```
            {
                Console.WriteLine($"{DateTime.Now}: Error al eliminar el
archivo - {ex.Message}");
            }
        }

        public void Start()
        {
            _timer.Start();
            Console.WriteLine("Servicio iniciado.");
        }

        public void Stop()
        {
            _timer.Stop();
            Console.WriteLine("Servicio detenido.");
        }
    }
}
```

Examen

Evaluación 7

Programador en C#

Tu calificación: 8

(8 de 10)

Repetir examen