Tutorial: Criar um aplicativo de serviço do Windows

Neste artigo

Aviso

Esta documentação não é para a versão mais recente do Windows Service. Para obter o conteúdo mais recente sobre os Serviços do Windows usando o **BackgroundService** e o modelo Serviço de Trabalho, consulte:

- **Serviços de trabalho no .NET**
- **Criar um serviço do Windows usando o BackgroundService**

Este artigo demonstra como criar um aplicativo de serviço do Windows no Visual Studio que grava mensagens em um log de eventos.

Criar um serviço

Para começar, crie o projeto e defina os valores necessários para que o serviço funcione corretamente.

1. No menu **Arquivo** do Visual Studio, selecione Novo Projeto > (ou pressione ++) para abrir a janela **Novo Projeto**. `CtrlShiftN`
2. Localize e selecione o modelo **de projeto Serviço do Windows (.NET Framework).**

   Nota

   Se você não vir o modelo **de serviço do Windows**, talvez seja necessário instalar a carga de trabalho de **desenvolvimento de área de trabalho do .NET** usando o Visual Studio Installer.

3. Em **Nome**, digite *MyNewService* e selecione **OK.**

   A guia Design aparece (Service1.cs [Design] ou **Service1.vb [Design]**).

O modelo de projeto inclui uma classe de componente chamada que herda de System.ServiceProcess.ServiceBase. Ele inclui grande parte do código de serviço básico, como o código para iniciar o serviço.`Service1`

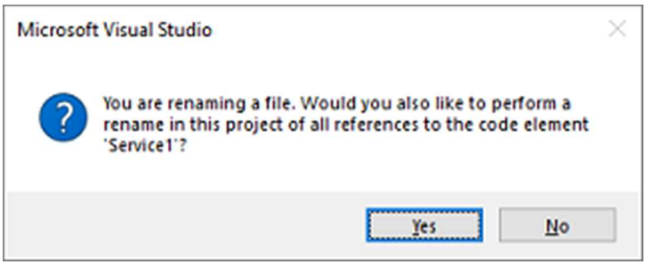
Renomear o serviço

Renomeie o serviço de **Service1** para **MyNewService**.

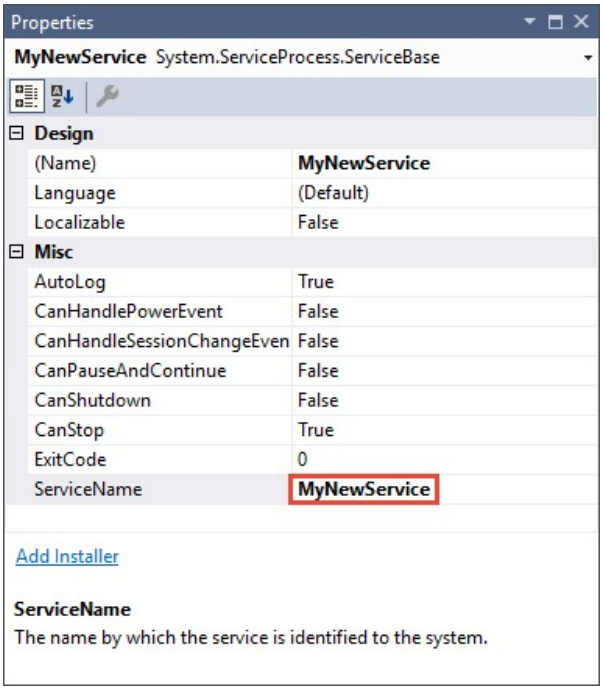
1. No **Gerenciador de Soluções**, selecione Service1.cs ou **Service1.vb** e escolha **Renomear** no menu de atalho. Renomeie o arquivo para MyNewService**.cs ou MyNewService.vb** e pressione `Enter`

   Uma janela pop-up é exibida perguntando se você deseja renomear todas as referências ao elemento de código *Service1*.

2. Na janela pop-up, selecione **Sim**.



3. Na guia **Design**, selecione **Propriedades** no menu de atalho. Na janela **Propriedades**, altere o valor **ServiceName** para *MyNewService*.



   Selecione **Salvar tudo** no menu **Arquivo**.

Adicionar recursos ao serviço

Nesta seção, você adiciona um log de eventos personalizado ao serviço do Windows. O componente EventLog é um exemplo do tipo de componente que você pode adicionar a um serviço do Windows.

Adicionar funcionalidade de log de eventos personalizada

1. No **Gerenciador de Soluções**, no menu de atalho para MyNewService.cs ou **MyNewService.vb**, escolha **Exibir Designer**.
2. Na **Caixa de Ferramentas**, expanda **Componentes** e arraste o componente **EventLog** para a guia Service1.cs [Design**] ou Service1.vb [Design].**
3. No **Gerenciador de Soluções**, no menu de atalho para MyNewService.cs ou **MyNewService.vb**, escolha **Exibir código**.
4. Defina um log de eventos personalizado.

   Para C#, edite o construtor existente conforme mostrado no trecho de código a seguir. Para Visual Basic, adicione o construtor conforme mostrado no trecho de código a seguir.`MyNewService()New()`

   C#Copiar

   ```
   public MyNewService()
   {
       InitializeComponent();
       eventLog1 = new System.Diagnostics.EventLog();
       if (!System.Diagnostics.EventLog.SourceExists("MySource"))
       {
           System.Diagnostics.EventLog.CreateEventSource(
               "MySource","MyNewLog");
       }
       eventLog1.Source = "MySource";
       eventLog1.Log = "MyNewLog";
   }
   ```

5. Adicione uma instrução a MyNewService.cs (se ainda não existir) ou uma instrução a **MyNewService.vb**, para o namespace System.Diagnostics:`usingImports`

   C#Copiar

   ```
   using System.Diagnostics;
   ```

6. Selecione **Salvar tudo** no menu **Arquivo**.

Definir o que ocorre quando o serviço é iniciado

No editor de código para MyNewService**.cs ou MyNewService.vb**, localize o método OnStart. Visual Studio criou automaticamente uma definição de método vazia quando você criou o projeto. Adicione código que grava uma entrada no log de eventos quando o serviço é iniciado:

C#Copiar

```
protected override void OnStart(string[] args)
{
    eventLog1.WriteEntry("In OnStart.");
}
```

Votação

Como um aplicativo de serviço foi projetado para ser de longa execução, ele geralmente sonda ou monitora o sistema, que você configurou no método OnStart. O método deve retornar ao sistema operacional após o início da operação do serviço para que o sistema não seja bloqueado.`OnStart`

Para configurar um mecanismo de sondagem simples, use o componente System.Timers.Timer. O temporizador gera um evento Decorrido em intervalos regulares, momento em que seu serviço pode fazer seu monitoramento. Use o componente Timer da seguinte maneira:

- Defina as propriedades do componente Timer no método.`MyNewService.OnStart`
- Inicie o temporizador chamando o método Start.

Configurar o mecanismo de sondagem

1. Adicione uma instrução a MyNewService.cs ou uma instrução a **MyNewService.vb**, para o namespace System.Timers:`usingImports`

   C#Copiar

   ```
   using System.Timers;
   ```

2. Adicione o seguinte código no evento para configurar o mecanismo de sondagem:`MyNewService.OnStart`

   C#Copiar

   ```
   // Set up a timer that triggers every minute.
   Timer timer = new Timer();
   timer.Interval = 60000; // 60 seconds
   timer.Elapsed += new ElapsedEventHandler(this.OnTimer);
   timer.Start();
   ```

3. Na classe, adicione uma variável de membro. Ele contém o identificador do próximo evento a ser gravado no log de eventos:`MyNewService`

   C#Copiar

   ```
   private int eventId = 1;
   ```

4. Na classe, adicione o método para manipular o evento Timer.Elapsed:`MyNewServiceOnTimer`

C#Copiar

```
public void OnTimer(object sender, ElapsedEventArgs args)
{
    // TODO: Insert monitoring activities here.
    eventLog1.WriteEntry("Monitoring the System",
EventLogEntryType.Information, eventId++);
}
```

Em vez de executar todo o seu trabalho no thread principal, você pode executar tarefas usando threads de trabalho em segundo plano. Para obter mais informações, consulte System.ComponentModel.BackgroundWorker.

Definir o que ocorre quando o serviço é interrompido

Insira uma linha de código no método OnStop que adiciona uma entrada ao log de eventos quando o serviço é interrompido:

C#Copiar

```
protected override void OnStop()
{
    eventLog1.WriteEntry("In OnStop.");
}
```

Definir outras ações para o serviço

Você pode substituir os métodos OnPause, OnContinue e OnShutdown para definir processamento adicional para seu componente.

O código a seguir mostra como você pode substituir o método OnContinue na classe:`MyNewService`

C#Copiar

```
protected override void OnContinue()
{
    eventLog1.WriteEntry("In OnContinue.");
}
```

Definir status do serviço

Os serviços relatam seu status ao Gerenciador de Controle de Serviços para que um usuário possa saber se um serviço está funcionando corretamente. Por padrão, um serviço que herda do ServiceBase relata um conjunto limitado de configurações de status, que incluem SERVICE_STOPPED, SERVICE_PAUSED e SERVICE_RUNNING. Se um serviço demorar um pouco para ser iniciado, é útil relatar um status de SERVICE_START_PENDING.

Você pode implementar as configurações de status de SERVICE_START_PENDING e SERVICE_STOP_PENDING adicionando código que chama a função SetServiceStatus do Windows.

Implementar status pendente de serviço

1. Adicione uma instrução a MyNewService.cs ou uma instrução a **MyNewService.vb**, para o namespace System.Runtime.InteropServices:`usingImports`

   C#Copiar

   ```
   using System.Runtime.InteropServices;
   ```

2. Adicione o seguinte código a MyNewService**.cs ou MyNewService.vb**, para declarar os valores e adicionar uma estrutura para o status, que você usará em uma chamada de chamada de invocação de plataforma:`ServiceState`

   C#Copiar

   ```
   public enum ServiceState
   {
       SERVICE_STOPPED = 0x00000001,
       SERVICE_START_PENDING = 0x00000002,
       SERVICE_STOP_PENDING = 0x00000003,
       SERVICE_RUNNING = 0x00000004,
       SERVICE_CONTINUE_PENDING = 0x00000005,
       SERVICE_PAUSE_PENDING = 0x00000006,
       SERVICE_PAUSED = 0x00000007,
   }

   [StructLayout(LayoutKind.Sequential)]
   public struct ServiceStatus
   {
       public int dwServiceType;
       public ServiceState dwCurrentState;
       public int dwControlsAccepted;
       public int dwWin32ExitCode;
       public int dwServiceSpecificExitCode;
       public int dwCheckPoint;
       public int dwWaitHint;
   };
   ```

   Nota

   O Gerenciador de Controle de Serviço usa os membros e da **estrutura SERVICE_STATUS** para determinar quanto tempo esperar para que um serviço do Windows seja iniciado ou desligado. Se os métodos e forem longos, o serviço poderá solicitar mais tempo chamando novamente com um valor incrementado.`dwWaitHintdwCheckpointOnStartOnStopSetServiceStatusdwCheckPoint`

3. Na classe, declare a função SetServiceStatus usando a invocação da plataforma:`MyNewService`

C#Copiar

```
[DllImport("advapi32.dll", SetLastError = true)]
private static extern bool SetServiceStatus(System.IntPtr handle, ref
ServiceStatus serviceStatus);
```

4. Para implementar o status SERVICE_START_PENDING, adicione o seguinte código ao início do método OnStart:

C#Copiar

```
// Update the service state to Start Pending.
ServiceStatus serviceStatus = new ServiceStatus();
serviceStatus.dwCurrentState = ServiceState.SERVICE_START_PENDING;
serviceStatus.dwWaitHint = 100000;
SetServiceStatus(this.ServiceHandle, ref serviceStatus);
```

5. Adicione código ao final do método para definir o status como SERVICE_RUNNING:`OnStart`

C#Copiar

```
// Update the service state to Running.
serviceStatus.dwCurrentState = ServiceState.SERVICE_RUNNING;
SetServiceStatus(this.ServiceHandle, ref serviceStatus);
```

6. (Opcional) Se OnStop for um método de execução longa, repita esse procedimento no método. Implemente o status do SERVICE_STOP_PENDING e retorne o status do SERVICE_STOPPED antes que o método saia.`OnStop``OnStop`

   Por exemplo:

C#Copiar

```
// Update the service state to Stop Pending.
ServiceStatus serviceStatus = new ServiceStatus();
serviceStatus.dwCurrentState = ServiceState.SERVICE_STOP_PENDING;
serviceStatus.dwWaitHint = 100000;
SetServiceStatus(this.ServiceHandle, ref serviceStatus);

// Update the service state to Stopped.
serviceStatus.dwCurrentState = ServiceState.SERVICE_STOPPED;
SetServiceStatus(this.ServiceHandle, ref serviceStatus);
```

Adicionar instaladores ao serviço

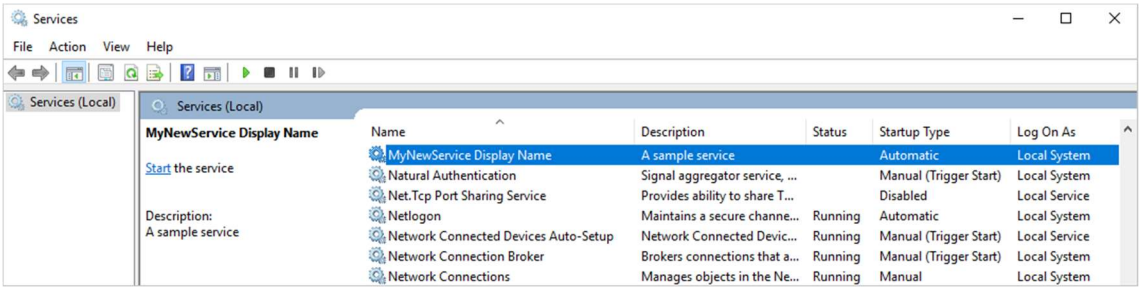
Antes de executar um serviço do Windows, você precisa instalá-lo, que o registra com o Gerenciador de Controle de Serviços. Adicione instaladores ao seu projeto para lidar com os detalhes de registro.

1. No **Gerenciador de Soluções**, no menu de atalho para MyNewService.cs ou **MyNewService.vb**, escolha **Exibir Designer**.
2. Na visualização **Design**, selecione a área de plano de fundo e escolha **Adicionar instalador** no menu de atalho.

Por padrão, o Visual Studio adiciona uma classe de componente chamada , que contém dois instaladores, ao seu projeto. Esses instaladores são para o seu serviço e para o processo associado ao serviço.`ProjectInstaller`

3. No modo **Design** para **ProjectInstaller**, selecione serviceInstaller1 para um projeto Visual C# ou **ServiceInstaller1** para um projeto Visual Basic e, em seguida, escolha **Propriedades** no menu de atalho.
4. Na janela **Propriedades**, verifique se a propriedade ServiceName está definida como **MyNewService**.
5. Adicione texto à propriedade Description, como *Um serviço de exemplo*.

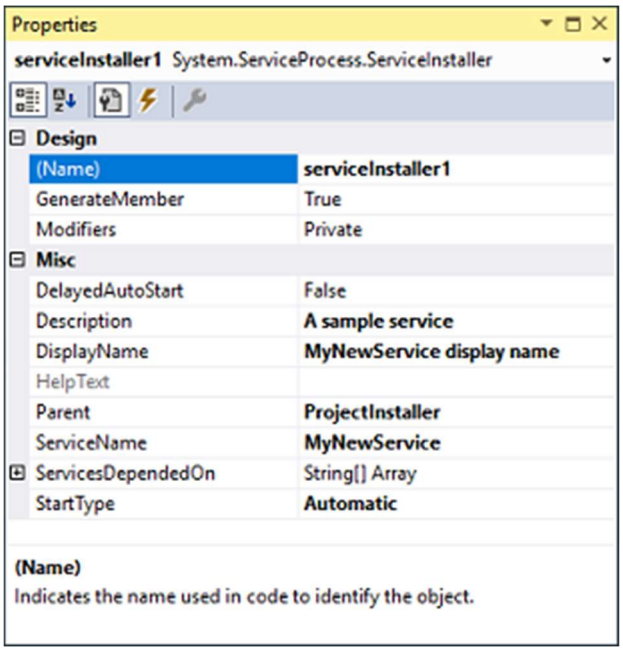
   Esse texto aparece na coluna **Descrição** da janela **Serviços** e descreve o serviço para o usuário.



6. Adicione texto à propriedade DisplayName. Por exemplo, *MyNewService Display Name*.

   Esse texto aparece na coluna **Nome para Exibição** da janela **Serviços**. Esse nome pode ser diferente da propriedade ServiceName, que é o nome que o sistema usa (por exemplo, o nome usado para o comando para iniciar o serviço).`net start`

7. Defina a propriedade StartType como Automatic na lista suspensa.
8. Quando terminar, as janelas **Propriedades** deverão ter a seguinte aparência:



9. No modo **Design** para **ProjectInstaller**, escolha serviceProcessInstaller1 para um projeto Visual C# ou **ServiceProcessInstaller1** para um projeto Visual Basic e, em seguida, escolha **Propriedades** no menu de atalho. Defina a propriedade Account como LocalSystem na lista suspensa.

   Essa configuração instala o serviço e o executa usando a conta do sistema local.

   Importante

   A conta **LocalSystem** tem permissões amplas, incluindo a capacidade de gravar no log de eventos. Use essa conta com cuidado, pois isso pode aumentar o risco de ataques de softwares mal-intencionados. Para outras tarefas, considere o uso da conta **LocalService**, que atua como um usuário sem privilégios no computador local e apresenta credenciais anônimas para qualquer servidor remoto. Este exemplo falhará se você tentar usar a conta **LocalService**, porque ela precisa de permissão para gravar no log de eventos.

Para obter mais informações sobre instaladores, consulte Como: Adicionar instaladores ao seu aplicativo de serviço.

(Opcional) Definir parâmetros de inicialização

Nota

Antes de decidir adicionar parâmetros de inicialização, considere se é a melhor maneira de passar informações para o seu serviço. Embora sejam fáceis de usar e analisar, e um usuário possa facilmente substituí-los, eles podem ser mais difíceis para um usuário descobrir e usar sem documentação. Geralmente, se o serviço requer mais do que apenas alguns parâmetros de inicialização, você deve usar o registro ou um arquivo de configuração.

A Windows service can accept command-line arguments, or startup parameters. When you add code to process startup parameters, a user can start your service with their own custom startup parameters in the service properties window. However, these startup parameters aren't persisted the next time the service starts. To set startup parameters permanently, set them in the registry.

Each Windows service has a registry entry under the **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services** subkey. Under each service's subkey, use the **Parameters** subkey to store information that your service can access. You can use application configuration files for a Windows service the same way you do for other types of programs. For sample code, see ConfigurationManager.AppSettings.

To add startup parameters

1. Select **Program.cs**, or **MyNewService.Designer.vb**, then choose **View Code** from the shortcut menu. In the method, change the code to add an input parameter and pass it to the service constructor:`Main`

C#Copy

```
static void Main(string[] args)
{
    ServiceBase[] ServicesToRun;
    ServicesToRun = new ServiceBase[]
    {
        new MyNewService(args)
    };
    ServiceBase.Run(ServicesToRun);
}
```

2. In **MyNewService.cs**, or **MyNewService.vb**, change the constructor to process the input parameter as follows:`MyNewService`

C#Copy

```
using System.Diagnostics;

public MyNewService(string[] args)
{
    InitializeComponent();

    string eventSourceName = "MySource";
    string logName = "MyNewLog";

    if (args.Length > 0)
    {
       eventSourceName = args[0];
    }

    if (args.Length > 1)
    {
        logName = args[1];
    }

    eventLog1 = new EventLog();

    if (!EventLog.SourceExists(eventSourceName))
    {
        EventLog.CreateEventSource(eventSourceName, logName);
    }

    eventLog1.Source = eventSourceName;
    eventLog1.Log = logName;
}
```

This code sets the event source and log name according to the startup parameters that the user supplies. If no arguments are supplied, it uses default values.

3. To specify the command-line arguments, add the following code to the class in **ProjectInstaller.cs**, or **ProjectInstaller.vb**:`ProjectInstaller`

C#Copy

```
protected override void OnBeforeInstall(IDictionary savedState)
{
    string parameter = "MySource1\" \"MyLogFile1";
    Context.Parameters["assemblypath"] = "\"" +
Context.Parameters["assemblypath"] + "\" \"" + parameter + "\"";
    base.OnBeforeInstall(savedState);
}
```

Typically, this value contains the full path to the executable for the Windows service. For the service to start up correctly, the user must supply quotation marks for the path and each individual parameter. A user can change the parameters in the **ImagePath** registry entry to change the startup parameters for the Windows service. However, a better way is to change the value programmatically and expose the functionality in a user-friendly way, such as by using a management or configuration utility.

Build the service

1. In **Solution Explorer**, choose **Properties** from the shortcut menu for the **MyNewService** project.

   The property pages for your project appear.

2. On the **Application** tab, in the **Startup object** list, choose **MyNewService.Program**, or **Sub Main** for Visual Basic projects.
3. To build the project, in **Solution Explorer**, choose **Build** from the shortcut menu for your project (or press ++).`CtrlShiftB`

Install the service

Now that you've built the Windows service, you can install it. To install a Windows service, you must have administrator credentials on the computer where it's installed.

1. Open Developer Command Prompt for Visual Studio with administrative credentials.
2. In **Developer Command Prompt for Visual Studio**, navigate to the folder that contains your project's output (by default, the *\bin\Debug* subdirectory of your project).
3. Enter the following command:

   shellCopy

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\installutil MyNewService.exe
```

   Se o serviço for instalado com êxito, o comando relatará êxito.

Se o sistema não conseguir localizar *installutil.exe, certifique-se* de que existe no computador. Essa ferramenta é instalada com o .NET Framework na pasta *%windir%\Microsoft.NET\Framework[64]\<framework version>*. Por exemplo, o caminho padrão para a versão de 64 bits é *%windir%\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe*.

Se o processo **installutil.exe** falhar, verifique o log de instalação para descobrir o motivo. Por padrão, o log está na mesma pasta que o executável do serviço. A instalação pode falhar se:

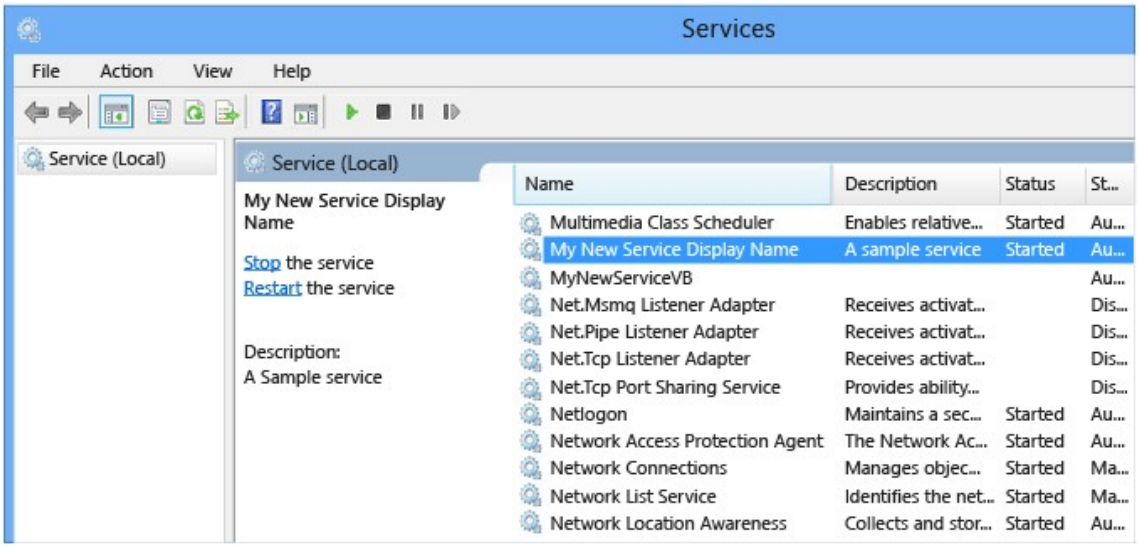
- A classe RunInstallerAttribute não está presente na classe.`ProjectInstaller`
- O atributo não está definido como .`true`
- A classe não é definida como .`ProjectInstallerpublic`

Para obter mais informações, consulte Como: Instalar e desinstalar serviços.

Iniciar e executar o serviço

1. No Windows, abra o aplicativo de área **de trabalho Serviços**.
   Pressione **Windows**+**R** para abrir a caixa **Executar**, digite *services.msc* e pressione ou selecione **OK**.`Enter`

   Você verá seu serviço listado em **Serviços**, exibido em ordem alfabética pelo nome de exibição definido para ele.



2. Para iniciar o serviço, escolha **Iniciar** no menu de atalho do serviço.
3. Para interromper o serviço, escolha **Parar** no menu de atalho do serviço.
4. (Opcional) Na linha de comando, use os comandos net start <nome do serviço> e **net stop <nome do serviço>** para iniciar e parar o **serviço**.

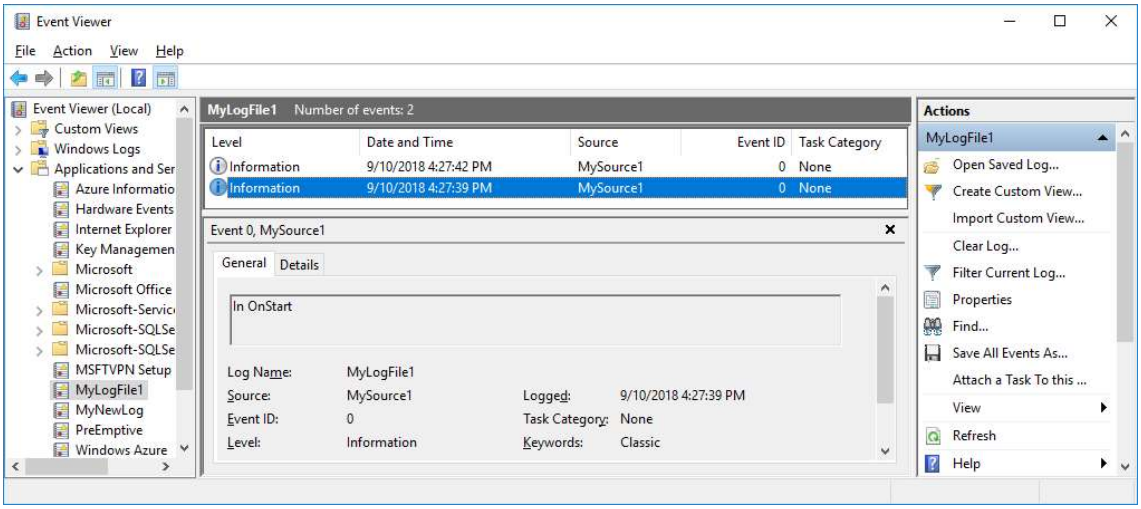
Verificar a saída do log de eventos do seu serviço

1. No Windows, abra o aplicativo de área de trabalho **Visualizador de Eventos**. Insira *Visualizar Eventos* na barra de pesquisa do Windows e selecione **Visualizar Eventos** nos resultados da pesquisa.

   Ponta

   No Visual Studio, você pode acessar logs de eventos abrindo o **Gerenciador de Servidores** no menu **Exibir** (ou pressione ++) e expandindo o nó **Logs de Eventos** para o computador local.`CtrlAltS`

2. Em **Visualizar Eventos**, **expanda Logs de Aplicativos e Serviços**.
3. Localize a listagem de **MyNewLog** (ou **MyLogFile1** se você seguiu o procedimento para adicionar argumentos de linha de comando) e expanda-a. Você deve ver as entradas para as duas ações (iniciar e parar) que seu serviço executou.



Clean up resources

If you no longer need the Windows service app, you can remove it.

1. Open **Developer Command Prompt for Visual Studio** with administrative credentials.
2. In the **Developer Command Prompt for Visual Studio** window, navigate to the folder that contains your project's output.
3. Enter the following command:

   shellCopy

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\installutil.exe /u MyNewService.exe
```

If the service uninstalls successfully, the command reports that your service was successfully removed. For more information, see How to: Install and uninstall services.

Próximos passos

Agora que você criou o serviço, você pode:

- Crie um programa de instalação autônomo para outras pessoas usarem para instalar seu serviço do Windows. Use o WiX Toolset para criar um instalador para um serviço do Windows. Para outras ideias, consulte Criar um pacote do instalador.
- Explore o componente ServiceController, que permite enviar comandos para o serviço que você instalou.
- Em vez de criar o log de eventos quando o aplicativo é executado, use um instalador para criar um log de eventos quando você instalar o aplicativo. O log de eventos é excluído pelo instalador quando você desinstala o aplicativo. Para obter mais informações, consulte EventLogInstaller.