

Windows servisi

(Windows) servisi

❑ Windows servisi (engl. *Windows Services*)

- dugotrajni (*long-running*) izvršni programi
- nemaju korisničkog sučelja, ne zahtijevaju interakciju s korisnikom
- izvršavaju se neovisno o prijavi korisnika na računalo
- mogu biti automatski pokrenuti pri pokretanju operacijskog sustava
- moraju biti instalirani na računalo da bi se mogli pokrenuti
- OS Unix imaju sličan tip programa pod nazivom demon (*daemon*)

❑ Životni vijek windows servisa – nekoliko internih stanja

- instalacija servisa na računalo - servis se učitava u *Services Control Manager* (centralno mjesto za administraciju servisa)
- pokretanje – nakon što je servis instaliran može se pokrenuti.
 - Pokretanje se obavlja iz *Management Console*, pozivom *Start* metode iz programskog koda, automatski prilikom pokretanja računala ...
- nakon pokretanja servis se nalazi u *Running* stanju dok ne bude zaustavljen (stanje *Stopped*), pauziran (*Paused*) ili dok računalo ne bude isključeno.

Upravljanje instaliranim servisima

☐ Upravljanje servisom vrši se kroz *Services Management Console*

- Control Panel \ Administrative Tools \ Services ili
- My Computer – Manage \ Services and Applications \ Services

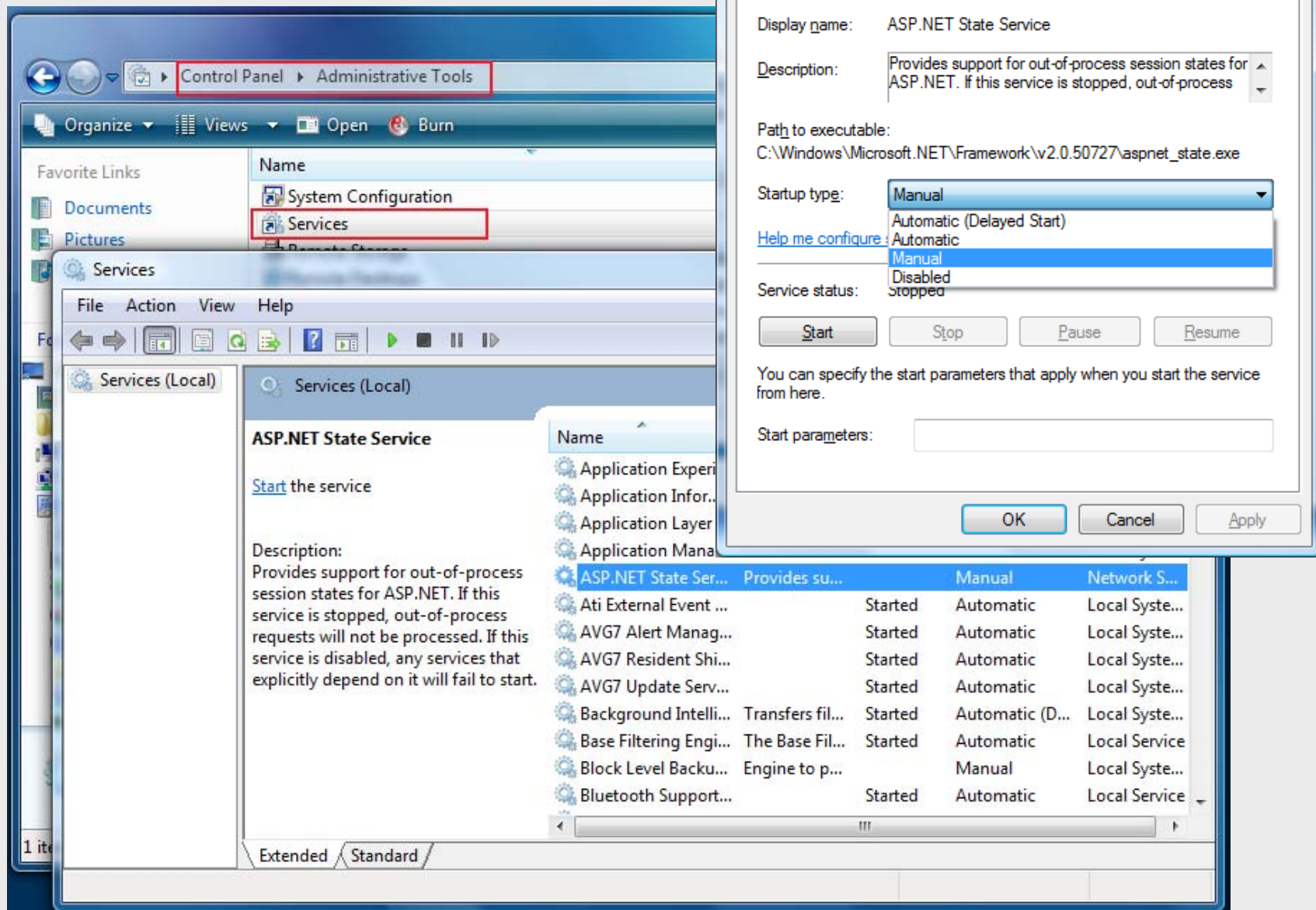
☐ Pregled postojećih servisa na računalu i njihovo stanje

- pokretanje, zaustavljanje, pauziranje servisa
- administriranje načina pokretanja servisa
 - automatski – servis se pokreće pri pokretanju OS
 - automatski (odgođeno) – nakon pokretanja OS kada završe početne akcije koje zauzimaju procesne resurse
 - ručno – pokreće korisnik iz konzole ili drugi program *Start* metodom
 - onemogućeno

☐ Iako servisi obično nemaju korisničko sučelje, programer može dodati formu ili neku drugu UI komponentu.

- U tom slučaju je potrebno označiti “*Allow service to interact with desktop*” u *Logon* dijelu panela s postavkama.

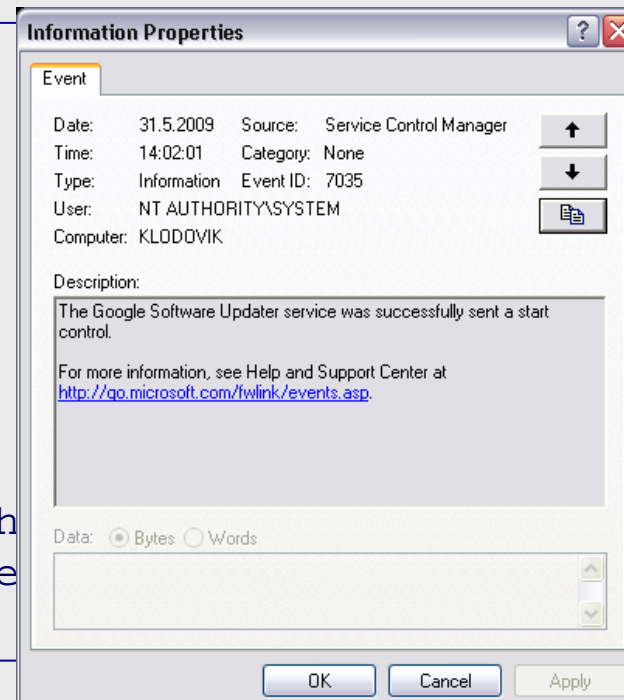
Upravljanje servisima



Informacije o radu servisa

- ❑ Windows servisi nemaju korisničko sučelje pa rezultate izvođenja evidentiraju na drugačiji način u odnosu na obične aplikacije.
- ❑ Osnovne informacije o promjeni stanja servisa i njihovim akcijama zapisuju se u dnevnik događaja (*event log*)
 - pregled dnevnika obavlja se u dijelu konzole *Event Viewer*
 - Pokretanje: Control Panel \ Administrative Tools \ Event Viewer
 - preglednik evidentira i druge događaje te podatke koje su zapisale (logirale) i druge aplikacije te sam operacijski sustav.

Event Type: Information
Event Source: Service Control Manager
Event Category: None
Event ID: 7036
Date: 21.1.2008
Time: 22:30:01
User: N/A
Computer: KLODOVIK
Description:
The World Wide Web Publishing service entered th
For more information, see Help and Support Cente
<http://go.microsoft.com/fwlink/events.asp>.



Informacije o radu servisa

The screenshot shows a Windows XP desktop with the following elements:

- Control Panel > Administrative Tools:** The breadcrumb path at the top of the window.
- Computer Management:** The main window title and the icon in the left sidebar.
- Left Sidebar:** A tree view showing the hierarchy of system tools. The 'Application' log under 'Windows Logs' is selected and highlighted with a red box.
- Event List:** A table of system events. The event with ID 8224 from the VSS source is highlighted in blue.
- Event Details:** A pop-up window for 'Event 8224, VSS' showing the message: 'The VSS service is shutting down due to idle timeout.' It also displays metadata like Log Name, Source, Event ID, and Task Category.
- Actions Panel:** A list of actions available for the selected event, such as 'Open Saved Log...', 'Create Custom View...', and 'Event Properties'.

Level	Date and Time	Source	Event ID	Task Cat
Information	14.1.2008 10:23:23	MsiInstaller	11707	None
Information	14.1.2008 10:23:18	RestartManager	10001	None
Information	14.1.2008 10:23:18	System Restore	8194	None
Information	14.1.2008 10:23:12	RestartManager	10000	None
Information	14.1.2008 10:23:09	System Restore	8194	None
Information	14.1.2008 9:29:01	Outlook	32	None
Information	14.1.2008 9:29:00	Outlook	32	None
Information	14.1.2008 9:29:00	Outlook	32	None
Information	14.1.2008 9:29:00	Outlook	32	None
Information	14.1.2008 9:04:34	CertificateServi...	1	None
Information	14.1.2008 9:04:35	Winlogon	4101	None
Information	14.1.2008 0:03:10	VSS	8224	None
Information	14.1.2008 0:00:10	System Restore	8211	None

Event 8224, VSS

General Details

The VSS service is shutting down due to idle timeout.

Log Name: Application
Source: VSS
Event ID: 8224
Logged: 14.1.2008 0:03:10
Task Category: None

Izrada Windows servisa

❑ Izrada windows servisa u okolini *Visual Studio*

- prilikom kreiranja novog projekta odabire se predložak *Windows Service* koji automatski u projekt dodaje odgovarajući razred
- dodani razred nasljeđuje `System.ServiceProcess.ServiceBase` razred te je moguće implementirati ponašanje servisa pisanjem preopterećenih postupaka (`OnStart`, `OnStop`, `OnPause`, `OnContinue`,...).
- `ServiceBase` razred nalazi se u *System.ServiceProcess* knjižnici pa je referenca na knjižnicu također automatski dodana

```
namespace ZPR.ServiceDemo
{
    public partial class NadzornikServis: ServiceBase
    {
        ....
        public NadzornikServis()
```

❑ Primjer: **WinServis\NadzornikServis – NadzornikServis.cs**

- Servis za praćenje promjena (brisanja, dodavanja, izmjene datoteka i direktorija) u zadanom direktoriju

Izrada Windows servisa

❑ Primjer: WinServis\NadzornikServis – Program.cs

- Main postupak mora izdati Run naredbu svim servisima koje projekt sadrži.
- Pozivom Run metode servis se učitava u *Services Management Console*

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

- ## ❑ Ukoliko je windows servis projekt nastao korištenjem predloška odgovarajući kod u Main postupak je automatski dodan.

Programska arhitektura Windows servisa

❑ Ponašanje servisa definira se preko događaja osnovnog razreda `System.ServiceProcess.ServiceBase`

- `OnStart` – izvršava se prilikom pokretanja servisa
- `OnPause` – prilikom pauziranja servisa
- `OnStop` – prilikom zaustavljanja servisa
- `OnContinue` – pri povratku iz stanja zaustavljanja u stanje izvršavanja
- `OnShutdown` – prilikom isključivanja sustava
- `OnCustomCommand` – prilikom zadavanja korisničke naredbe (prima *int*)
- `OnPowerEvent` – prilikom power management događaja (npr. baterija slaba, suspend)

❑ Funkcionalnost servisa započinje kodom u `OnStart`

- (u našem primjeru kod za pokretanje nadgledanja direktorija)

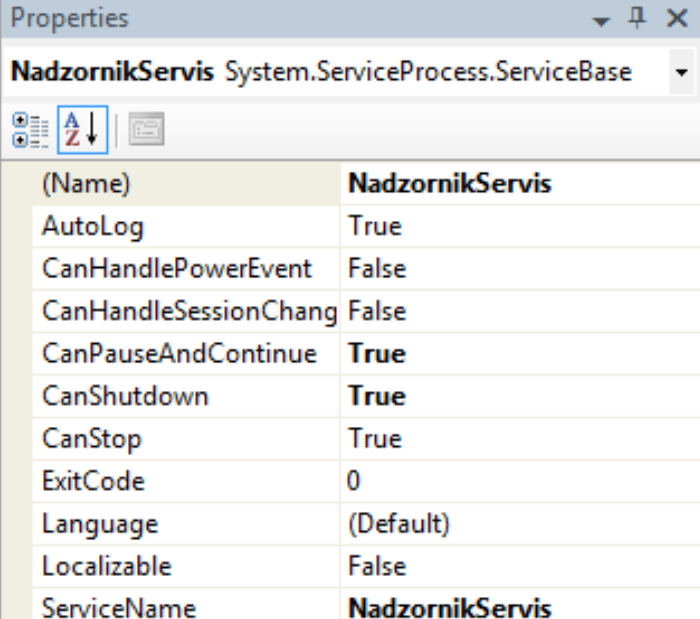
Programska arhitektura windows servisa

❑ Svojstva `System.ServiceProcess.ServiceBase` koja definiraju ponašanje servisa:

- `AutoLog` – automatsko zapisivanje u event log uobičajenih događaja (npr. *Install* ili *Start*)
- `CanHandlePowerEvent` – da li servis obrađuje *Power management* događaje (npr. baterija slaba, *suspend*)
- `CanStop` – mogućnost poziva zaustavljanja servisa (metoda `OnStop`)
- `CanPauseAndContinue` – mogućnost poziva `OnPause` i `OnContinue` metoda (pauziranje i nastavak izvršavanja servisa)
- `CanShutdown` – mogućnost izvršavanja `OnShutdown` metode na događaj isključivanja računala

❑ Primjer: **WinServis\NadzornikServis** – **NadzornikServis.cs**

- View Designer \ *Properties* , postaviti na *true* `CanPauseAndContinue`, `CanShutdown`, `CanStop`



The screenshot shows the 'Properties' window in Visual Studio for the project 'NadzornikServis'. The 'System.ServiceProcess.ServiceBase' properties are displayed. The 'CanPauseAndContinue', 'CanShutdown', and 'CanStop' properties are set to 'True', while 'AutoLog' is 'True' and 'CanHandlePowerEvent' is 'False'.

(Name)	NadzornikServis
AutoLog	True
CanHandlePowerEvent	False
CanHandleSessionChange	False
CanPauseAndContinue	True
CanShutdown	True
CanStop	True
ExitCode	0
Language	(Default)
Localizable	False
ServiceName	NadzornikServis

Razred FileSystemWatcher

❑ Razred FileSystemWatcher

- nadzire sustav datoteka (*file system*) i podiže događaje kada se dogodi promjena u nadziranom kazalu ili datoteci.

❑ Svojstva

- `Path` – putanja do nadziranog direktorija ili datoteke
- `Filter` – određuje što će se nadzirati u direktoriju npr. `*.*` ili `*.exe`
- `EnableRaisingEvents` - određuje da li je komponenta aktivna
- `NotifyFilter` – enumerator tipa `NotifyFilters` koji određuje koje promjene se prate npr. promjene u nazivu datoteke, vremenu zadnjeg pisanja u datoteku (`FileName`, `DirectoryName`, `LastWrite`,...)

❑ Događaji

- (`FileSystemEventHandler d`, `FileSystemEventArgs e`)
- `Changed` – podignut u slučaju promjene nad nadziranim kazalom ili datotekom
- `Created` – u slučaju kreiranja novog direktorija ili datoteke
- `Deleted` - u slučaju brisanja

Razred FileSystemWatcher

❑ Događaji

- (RenamedEventHandler d, RenamedEventArgs e)
- Renamed – u slučaju preimenovanja direktorija ili datoteke

❑ Razredi FileSystemEventArgs i RenamedEventArgs nasljeđuju razred EventArgs koji je bazni razred za razrede koji sadrže podatke o događajima.

❑ Svojstva FileSystemEventArgs razreda:

- ChangeType – tip događaja koji se dogodio
- FullPath – puna putanja do promijenjenog direktorija ili datoteke
- Name – ime promijenjenog direktorija ili datoteke

❑ Svojstva RenamedEventArgs razreda:

- ChangeType – tip događaja koji se dogodio
- FullPath – nova puna putanja do preimenovanog direktorija ili datoteke
- Name – novo ime preimenovanog direktorija ili datoteke
- OldFullPath – stara puna putanja do preimenovanog direktorija ili datoteke
- OldName – staro ime preimenovanog direktorija

Realizacija događaja servisa *nadzornik*

❑ Primjer: WinServis\NadzornikServis – NadzornikServis.cs

- Događaji prilikom kojih se informacije o promjeni zapisuju u dnevnik OS
- Pogledati događaje komponente `fileSystemWatcher` u dizajnu
- Razred `EventLog` definiran je u `SystemDiagnostics`


```
private void fileSystemWatcher1_Changed(  
    object s, System.IO.FileSystemEventArgs e)  
{  
    EventLog.WriteEntry("NadzornikServis: "  
        + e.FullPath + " " + e.ChangeType.ToString());  
}
```

```
private void fileSystemWatcher1_Renamed(object s,  
    System.IO.RenamedEventArgs e)  
{  
    EventLog.WriteEntry("NadzornikServis: " +  
        e.OldFullPath + " RENAMED TO " + e.FullPath);  
}
```

Konfiguracijska datoteka

- ❑ U datoteku `App.config` zapisana je putanja do direktorija koji će windows servis pratiti (*Path*):

```
<appSettings>
  <add key="Path" value="c:\\" />
</appSettings>
```

- ❑ **Primjer:**  **WinServis\NazdzornikServis – NazdzornikServis.cs**
 - pri pokretanju čita parametre i aktivira `FileSystemWatcher` objekt

```
private void SetPath()
{
    ConfigurationManager.RefreshSection("appSettings");
    fileSystemWatcher1.Path =
        ConfigurationManager.AppSettings["Path"].ToString();
    EventLog.WriteEntry("NadzornikServis: WATCHING-"
                        + fileSystemWatcher1.Path);
}
```


Događaji servisa - OnStart, OnStop

Primjer: WinServis\NadzornikServis – NadzornikServis.cs

- **OnStart** postupak servisa, dakle, postavlja praćeni direktorij, pokreće nadziranje te zapisuje vrijeme pokretanja u sistemski dnevnik (log)

```
protected override void OnStart(string[] args) {  
    isPaused = false;  
    SetPath();  
    fileSystemWatcher1.EnableRaisingEvents = true;  
    EventLog.WriteEntry("NadzornikServis : STARTED at " +  
        DateTime.Now.ToShortTimeString());  
}
```

- **OnStop** postupak zaustavlja praćenje direktorija i zapisuje u log

```
protected override void OnStop()  
{  
    fileSystemWatcher1.EnableRaisingEvents = false;  
    EventLog.WriteEntry("NadzornikServis: STOPPED at " +  
        DateTime.Now.ToShortTimeString());  
}
```

Događaji servisa – OnCustomCommand

❑ Omogućuje ugradnju korisnički definiranih postupaka

- Prima *int* parametar koji određuje koju radnju servis treba obaviti.
- Vrijednosti parametra od 0 do 127 su sistemski zauzete pa je za korisnički definirane komande dozvoljeno koristiti vrijednosti 128 do 256.

❑ Primjer: WinServis\NadzornikServis – NadzornikServis.cs

- npr. komanda 128 neka osvježava podatke o nadgledanom direktoriju iz konfiguracijske datoteke (iz NadzornikManager).

```
protected override void OnCustomCommand(int command)
{
    base.OnCustomCommand(command);
    if (command == 128)
    {
        SetPath();
    }
}
```

Ugradnja i izvođenje servisa

- ❑ Prevedena izvršna datoteka servisa mora biti instalirana da bi servis mogao obavljati željenu funkciju
- ❑ Nužno je izraditi instalacijsku komponentu.
 - Instalacijska komponenta instalira i registira servis na računalu te stvara ulaznu točku servisa za *Windows Service Control Manager*
- ❑ Servis se u razvojnoj okolini ne može pokretati ili *debugirati* na način uobičajen za interaktivne aplikacije, pritiskom gumba F5 ili F11 (*Visual Studio*)
- ❑ Tek nakon što se servis instalira i pokrene, može se koristiti *VS debugger* pri čemu se potrebno spojiti na proces servisa:
 - *Tools / Attach to Process*

Izrada instalacijske komponente

❑ Izrada instalacijske komponente

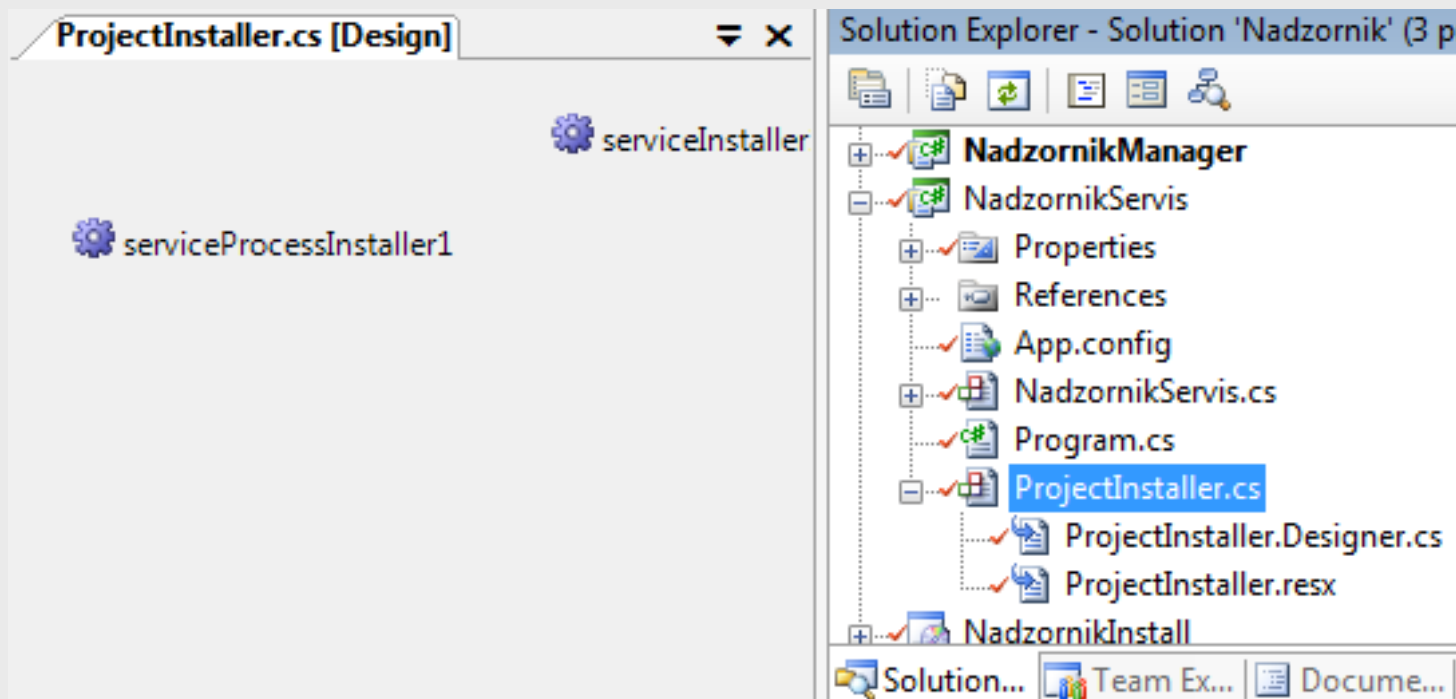
- *Add Installer* funkcija iz skočnog izbornika datoteke NadzornikServis.cs

❑ Primjer: WinServis\NazdzornikServis – NadzornikServis[Design].cs

The screenshot displays the Visual Studio IDE. On the left, the 'NadzornikServis.cs [Design]' window is open, showing a 'fileSystemWatcher1' control. A right-click context menu is visible over the design surface, with the 'Add Installer' option highlighted by a red rectangle. On the right, the 'Solution Explorer - Solution 'Nadzornik'' pane shows the project structure. The 'NadzornikServis' project is expanded, listing files: Properties, References, App.config, NadzornikServis.cs (highlighted with a red rectangle), NadzornikServis.Design, NadzornikServis.resx, Program.cs, and ProjectInstaller.cs. The bottom status bar shows 'Solution...', 'Team Ex...', and 'Docum'.

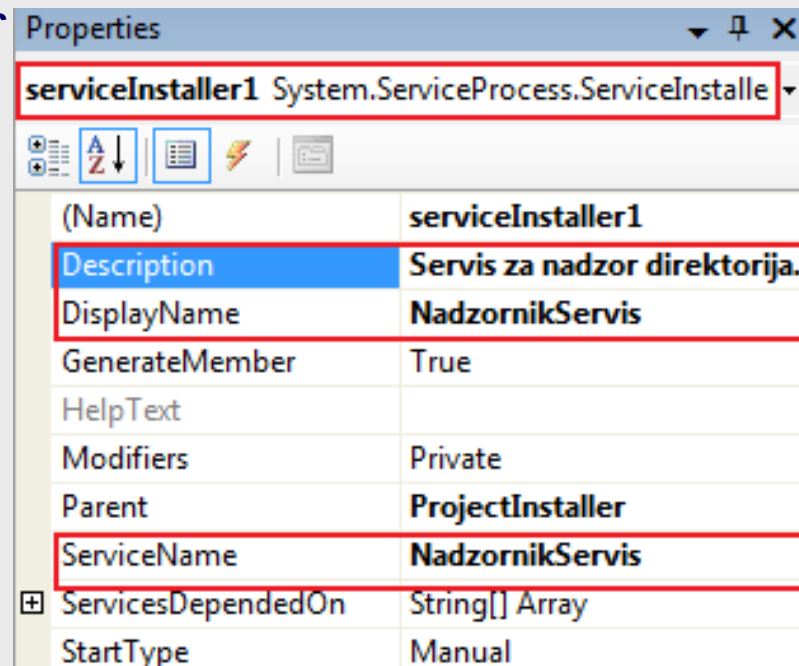
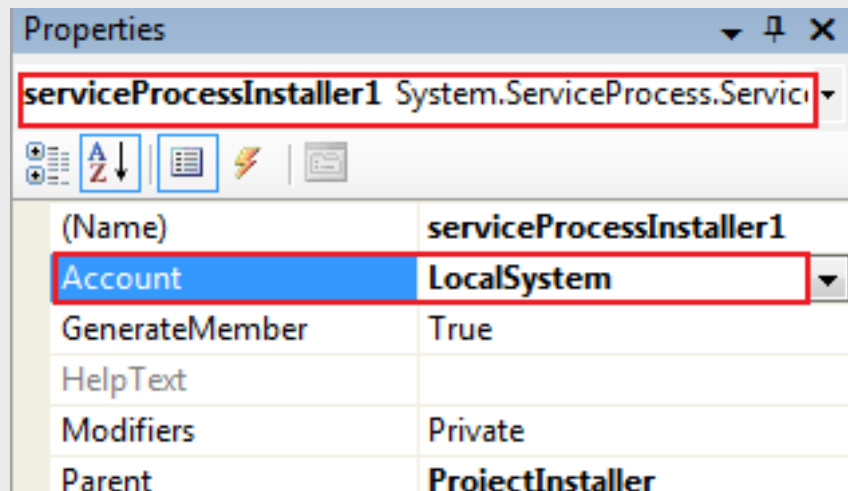
Instalacija servisa – ProjectInstaller.cs

- ❑ **Add Installer** funkcija dodaje u projekt datoteku `ProjectInstaller.cs` koja sadrži dvije komponente:
 - `serviceInstaller`
 - `serviceProcesInstaller`
- ❑ **Automatski je dodana i referenca projekta na `System.Configuration.Install` knjižnicu s navedenim razredima**



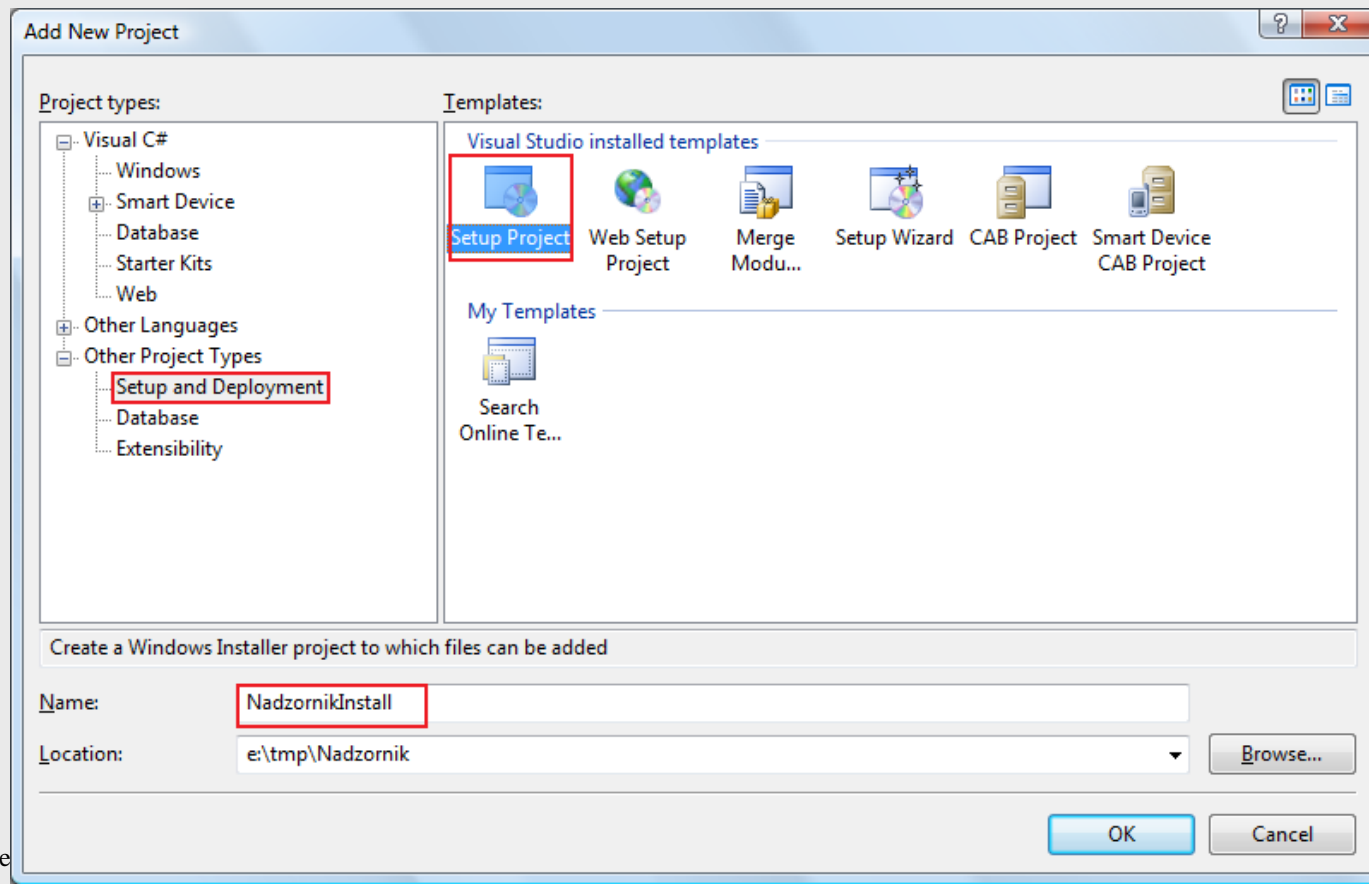
Instalacija servisa – ProjectInstaller.cs

- ❑ Objekti tipa `serviceInstaller` i `serviceProcesInstaller` pozvani su tijekom instalacije servisa.
- ❑ Svojstva razreda `serviceInstaller`
 - `Description` – opis servisa
 - `DisplayName` – ime servisa koje će se prikazati korisniku (*friendly name*)
 - `ServiceName` – ime servisa po kojem OS identificirati servis. Obavezno mora biti jednako imenu razreda koji je naslijedio `ServiceBase`
- ❑ Svojstva razreda `serviceInstaller`
 - `Account` – tip korisničkog računa pod kojim se servis izvršava



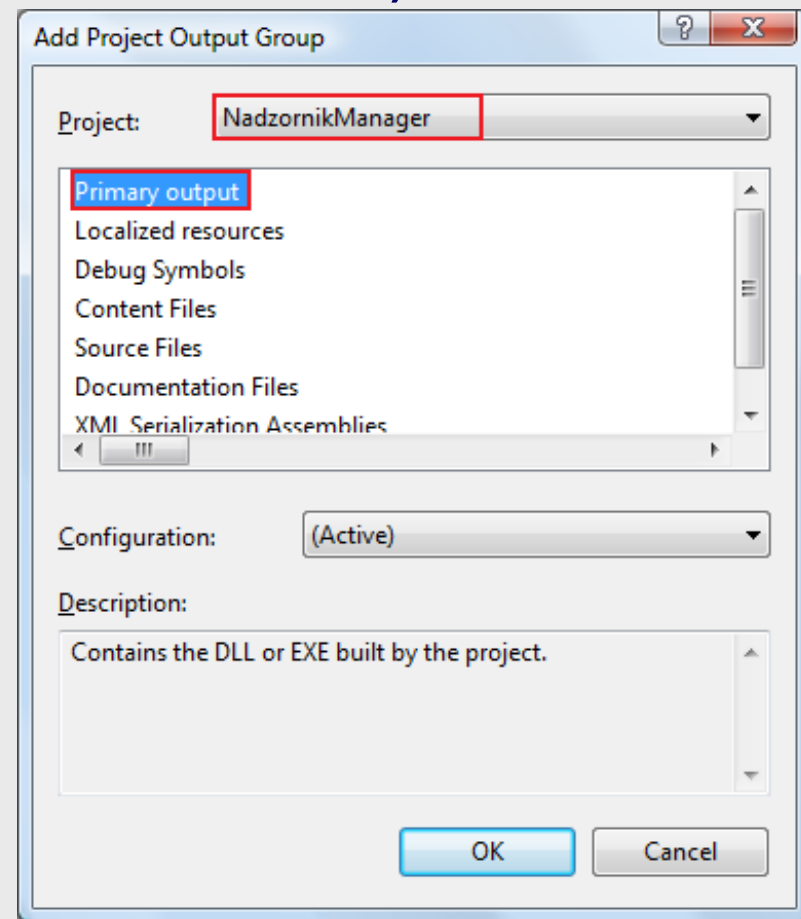
Instalacija servisa – instalacijski projekt

- ❑ Osim dodavanja *installer* datoteke za servis, potrebno je dodati i instalacijski projekt koji kopira izvršne datoteke u instalacijski direktorij te instalira servis na računalo
- ❑ Dodavanje instalacijskog projekta (*Add → New Project* u kontekstnom izborniku *Solution*):



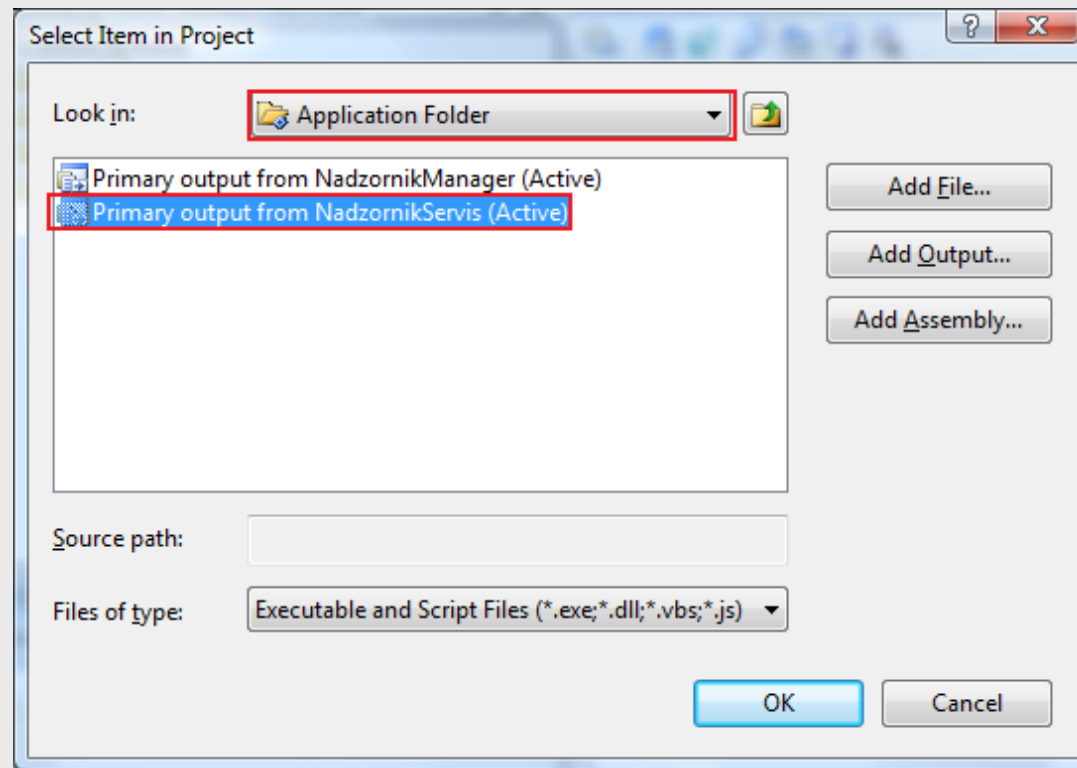
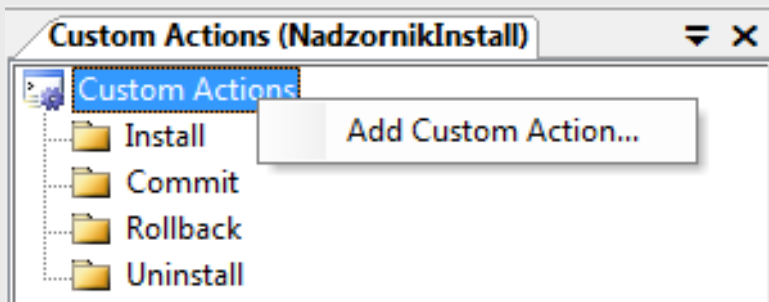
Instalacija servisa – instalacijski projekt

- ❑ U Instalacijskom projektu definiraju se radnje koje se izvršavaju tijekom instalacije kao što su kopiranje datoteka na konačnu lokaciju, postavljanje ikona i prečaca, provjera i instalacija potrebnih aplikacija i knjižnica (npr. .NET Framework).
- ❑ Za instaliranje servisa potrebno je dodati kompiliranu inačicu servisa u instalacijski projekt (*Primary output*):
 - skočni izbornik *NadzornikInstall* projekta (desni klik): *Add* → *Project Output*
 - u dobivenom dijalogu odabere se *NadzornikServis* projekt, odnosno njegov izlaz:



Instalacija servisa – instalacijski projekt

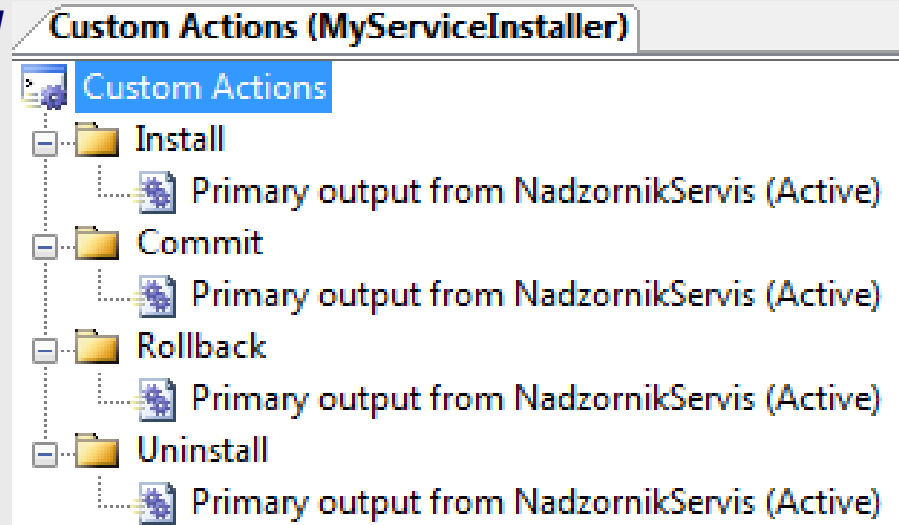
- ❑ Osim dodavanja kompilirane inačice (*Primary output*) potrebno je definirati i korisnički definirane akcije (*Custom actions*) za projekt:
 - kontekstni izbornik *NadzornikInstall* projekta (desni klik): *View* → *Custom Actions*
 - u otvorenom dijalogu iz kontekstnog izbornika nad *Custom Actions* odabrati *Add Custom Action*
 - odabrati *Application Folder* te *Primary output* od servisa



Instalacija servisa – instalacijski projekt

☐ **Primary output** dodan je na sve točke korisničkih akcija:

- *Install, Commit, Rollback, Uninstall*



☐ Instalacijski projekt se sada može sagraditi (*build*) te instalirati.

- *Build* i *Install* komande nalaze se u kontekstnom izborniku projekta.

☐ **Napomena:** instalaciju servisa može obaviti samo korisnik s administratorskim dozvolama.

- Ako se instalacija radi pod *OS Vista* iz razvojne okoline, potrebno je pokrenuti kao administrator (*Run as administrator*).
- Isto vrijedi i za instalacijsku datoteku koja je izlaz instalacijskog projekta.

Primjer

- ☐ Instalacija servisa
- ☐ Pokretanje i zaustavljanje
- ☐ Promjena sadržaja praćenog direktorija
- ☐ Uvid u dnevnik događaja (Event Viewer)

Izrada aplikacije za upravljanje windows servisom

□ Primjer:

- Za NadzornikServis servis (iz prošlog primjera) izraditi upravljačku aplikaciju koja upravlja radom servisa (*Start, Stop, Pause, Continue*) te omogućava promjenu nadgledanog direktorija.
- Aplikacija ne treba imati formu već samo ikonu u Windows status traci za upravljanje servisom putem kontekstnog izbornika.

□ U *Solution* (postojeći, koji već sadrži servis i instalacijski projekt) dodati novi projekt tipa *Windows Application*.

□ Na formu aplikacije (*Form1.cs*) iz alatne trake (*toolbox*) dovući sljedeće kontrole: `NotifyIcon`, `ContextMenuStrip` i `FolderBrowserDialog`

Razred NotifyIcon

❑ Komponenta koja kreira ikonu programa u Windows status traci

❑ Svojstva

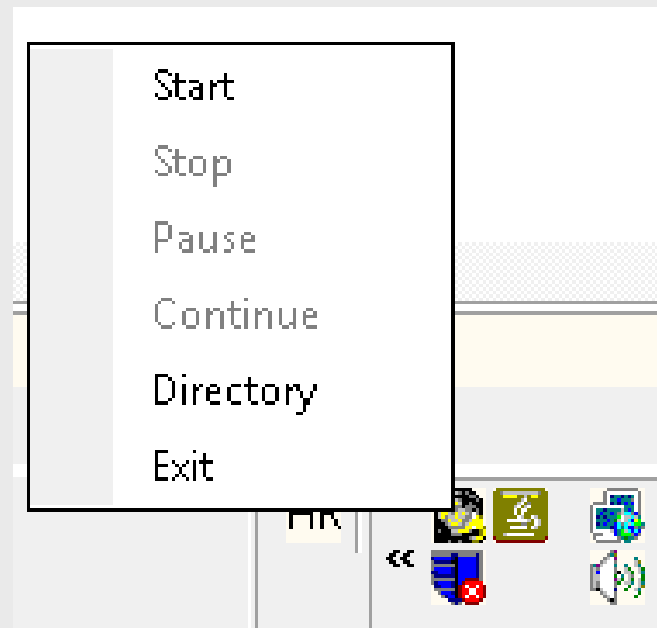
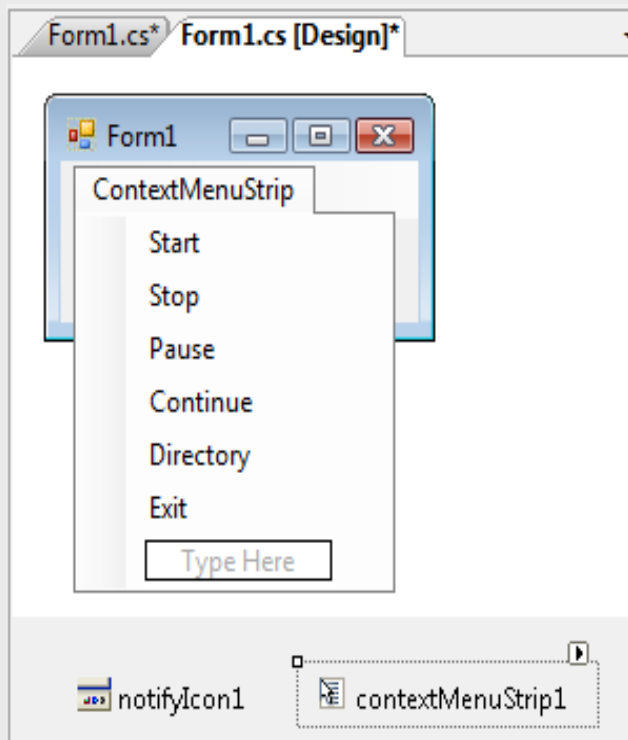
- `ContextMenu` – izbornik koji se prikazuje na desni klik ikone
- `Icon` – ikona komponente
- `Text` – tekst objašnjenja nad ikonom
- `Visible` – oznaka da će ikona biti prikazana
- `BalloonTipIcon` – ikona balončića vezanog uz ikonu
- `BalloonTipText` – tekst unutar balončića
- `BalloonTipTitle` – naslov tekst unutar balončića

❑ Događaji:

- `Click`, `DoubleClick`, `MouseMove`

Povezivanje NotifyIcon i ContextMenuStrip

❑ Primjer: WinServis\NadzornikManager – Form1.cs



❑ Primjer: NadzornikManager \ Form1.cs[Design]

- Povezivanje NotifyIcon kontrole i ContextMenuStrip-a – Svojstvo ContextMenuStrip u NotifyIcon objektu treba postaviti na ContextMenuStrip objekt koji smo dodali na formu

Razred `ServiceController`

❑ `ServiceController` razred omogućava:

- interakciju sa Windows servisima na lokalnom računalu i računalima na koje postoji pristup
 - dohvat dostupnih servisa
 - zadavanje naredbi za pokretanje, zaustavljanje, pauziranje servisa
 - zadavanje korisničkih (*custom*) naredbi

❑ Metode

- `GetServices` – dohvaća servise na računalu (`static` postupak)
- `Refresh` – osvježava vrijednosti svih svojstava
- `Start` – pokreće servis
- `Stop` – zaustavlja
- `Pause` – pauzira
- `Continue` – nastavlja

Primjer ServiceController

❏ Primjer: WinServis\NadzornikManager – Form1.cs

■ Upotreba ServiceController objekta

```
private ServiceController nadzornikServis = null;

private void CheckServiceInstallation()
{
    ServiceController[] installedServices;
    installedServices = ServiceController.GetService();
    foreach (ServiceController tmpService in installedServices)
    {
        if (tmpService.DisplayName == "NadzornikServis")
        {
            nadzornikServis = tmpService;
            return;
        }
    }
}
```

Primjer ServiceController

❑ Primjer: WinServis\NadzornikManager – Form1.cs

- Pokretanje procesa (prilikom klika na “*Start*” u kontekstnom izborniku):

```
private void startToolStripMenuItem_Click(  
    object s, EventArgs e){  
    try  
    {  
        nadzornikServis.Start();  
    }  
    catch (Exception ex) {}  
    finally  
    {  
        UpdateServiceStatus();  
    }  
}
```

Zapisivanje konfiguracije servisa

❏ Primjer: WinServis\NadzornikManager – Form1.cs

```
private void directoryToolStripMenuItem_Click(object sender,
EventArgs e)
{
    //mapiranje na konfiguracijsku datoteku drugog programa
    ExeConfigurationFileMap fileMap = new ExeConfigurationFileMap();
    fileMap.ExeConfigFilename = @"NadzornikServis.exe.config";
    System.Configuration.Configuration config =
        ConfigurationManager.OpenMappedExeConfiguration(fileMap,
            ConfigurationUserLevel.None);
    folderBrowserDialog1.SelectedPath =
        config.AppSettings.Settings["Path"].Value.ToString();
    folderBrowserDialog1.ShowDialog();
    config.AppSettings.Settings["Path"].Value=
        folderBrowserDialog1.SelectedPath.ToString();
    config.Save();
    if(nadzornikServis.Status==ServiceControllerStatus.Running)
        nadzornikServis.ExecuteCommand(128);
}
```




Zadaci za vježbu

- ❑ **Implementirati windows servis koji u zadanim vremenskim intervalima provjerava dnevnik OS-a:**
 - ukoliko su se u sistemskom logu, od zadnje provjere, pojavili zapisi o pogreškama servis treba poslati e-mail sa izvještajem o broju grešaka na zadane adrese.
 - e-mail adrese i vremenski interval zadaju se preko konfiguracijske datoteke servisa.

Reference

❑ MSDN dokumentacija o windows servisima

- [http://msdn2.microsoft.com/en-us/library/y817hyb6\(VS.80\).aspx](http://msdn2.microsoft.com/en-us/library/y817hyb6(VS.80).aspx)

❑ Wikipedija

- http://en.wikipedia.org/wiki/Windows_service