Tab 1

# Poglavlja

[**Poglavlja 1**](#_qqn77gn6dw8j)

[**Uvod 1**](#_80e53g5wqjih)

[**Windows Service .NET Framework 1**](#_c9m8vjxj2k8v)

[Općenito 1](#_9bjekotudqmi)

[Metode i stanja Servisa 2](#_nvo8iqvd0xpd)

[Stanja Servisa 2](#_qrhlhs6dtjaq)

[Metode 2](#_vgaf8y9r2g33)

[Installer 3](#_8civcklpn8wq)

[Logiranje 3](#_x9qpom994ke3)

[Instalacija i pokretanje 3](#_ahxax0o6jtqk)

# Uvod

Windows Servis se moze opisati kao software koji radi u “pozadini” nekakvu predefiniranu logiku. .NET Framework i .NET Core nam pruzaju *solution template* za izgradnju Windows Servisa sa vec podešenom “jezgrom” projekta tako da korisnik nakon sto napise logiku koja će se izvršavati ne mora dodatno konfigurirati sam solution ukoliko mu odgovaraju početne postavke. S obzirom da je .NET evoluirao kroz vrijeme tako da postoji određena razlika u .NET Framework i .NET Core rješenja za Windows Servis. U .NET Core-u zamjena za Windows Service je Worker Service koji se moze uz pomoć dodatnih paketa konfigurirati kao Windows Service.

Unutar Windows OS-a postoji mehanizam koji se zove *Task Scheduler*. Taj mehanizam ima sposobnost zakazivanja vremenskog interval za odrađivanje nekog zadatak, dok Windows Service odrađuje definirane zadatke cijelog svog zivotnog vijeka.

# Windows Service .NET Framework

## Općenito

Windows Servisi, u NET Framework-u, su software-i koji se instaliraju na računalu te odrađuju zadatke u pozadini, bez grafičkog sučelja te su odlična rješenja u slučaju da su ti zadaci *long-runnnig* procesi. Servis klasa, u kojoj je defnirana logika servisa, mora *inheritat* **ServiceBase** klasu.

Instalacija se vrši pomoću InstallUtil.exe, .NET Framework CLI instalacijskog alata, kojem kao argument dajemo lokaciju .exe datoteke koju dobijemo *builadnjem* projekta. Nakon što smo uspješno instalirali Windows Servis , dodatnu konfiguraciju te akcije kao što su pokretanje, zaustavljanje itd. radimo u *Services Control Manager-u.*

**Interakciju sa svim Windows Servisima možemo postići i kroz sam kod korištenjem *ServiceController* klase**

Da bi servis mogao instalirati na naš operacijski sustav potrebno je dodati *Installer* koji za nas izvrši instalaciju servisa.

Ovaj tip projekta se nemoze pokrenit kao recimo WebApi projekt (kroz Visual Studio) nego se mora instalirati i pokrenuti da bi vidili rezultate koda. Ova činjenica nam naročito utjeće na način na koji mozemo *debbugirat* ovaj tip projekta. Debbugiranje mozemo izvršiti na način da debugger “zakačimo” direktno na aktivni Windows Service process što moze dovest do neželjenog ponašanje posšto za vrijeme debbugiranja sam proces miruje i ukoliko smo pogodili *breakpoint* u debbuguru ocekivane akcije se nece izvršiti.

Također bi bilo bitno naglasiti da Windows Servis proces se pokreće unutat svog *security contexta* , što znaci da nije vezan uz trenutno ulogiranog User-a te mozemo definirati *permissione* za nas Servis unutar operacijskog sustava, bilo to da mu omogućimo Admin pristup ili neki drugi , više ograničeniji.

## Metode i stanja Servisa

### Stanja Servisa

Servis moze postojati u tri osnovna stanja a to su Running, Paused ili Stopped. Running stanje netreba dodatno objašnjvati , to je stanje kad je servis aktivan i obavlje svoj zadatak.

Pri pokretanju servisa (početak Running stanja) pokreće se Main metoda te kontruktor Servis klase. Tijekom tog “inicijalizacijskog” procesa možemo pohraniti neke podatke u memoriju. Ukoliko servis dođe u stanje Paused te se prebaci u Running stanje svi podaci će ostat zapamćeni za razliku od Stopped stanja koji resetira podatke. Ovo moze biti problem ukoliko je servis zadužen recimo za procesiranje Queue-a jer prilikom prelaska u Stopped stanje gubimo sve neprocesuirane podatke.

### Metode

Uvjet za Servis klasu je da inherita ServiceBase klasu. Ta bazna klasa nam omogucuje da definiramo metoda koje nam sa kojima definiramo ponašanje našeg servisa prilikom određenih događaja.

Ove metoda se pozivaju prilikom prelaska u jedno od tri osnovna stanja servisa. **OnStart** metoda je dobro mjesto za definirat inicjalne podatke ili provjeit jesu li svi uviejtu za rad servisa ispunjeni. **OnStop** metoda je dobro mjesto za defnirati *cleanup* logiku servisa kao što je brisanje file-ova ili zatvaranja konekcije sa bazom podataka itd.

**OnPause** metoda se izvrši netom prije nego se nas servis pauzira.

**OnShutdown** metoda se izvrši netom prije nego se ugasi računalo

**OnPowerEvent** metoda se izvrši prilikom notifikacije od strane sustava napajanje. U ovom slucaju mozemo zaustaviti servis ako je njegov utjecaj intezivan na bateriju.

Također mozemo definirati logiku za bilo koji *custom event* te je pozvati pomocu metode **OnCustomCommand.** Ova metoda prima parametar tipa integer u rasponu od 128 do 255 (sve ispod 128 je rezervirano za sistem) te za razne vrijednosti parametra mozemo izvršiti drugačiju logiku.

## Installer

U ovoj komponenti mozemo definirati osnovne podatke za *Services Control Manager* tablicu te security context pod kojim ce servis bit pokrenut.

Bitno je spomenuti da u Installeru definiramo *StartType* service :

* Manual - servis mora biti pokrenut ručno kroz CLI ili *Services Control Manager*
* Automatic - servis se pokreće sa pokretanjem OS-a

i Account za security context koji ce za ovaj primjer biti *LocalSystem*

## Logiranje

Ukoliko želimo vidjeti logove servisa koji nam mogu pomoci pri debugiranju ili ako samo želimo imati zapis o akcijama servisa to postižemo pomoću **EventLoger**klase*.*

Ukoliko nemamo potrebe za custom log-ovima dovoljno je postavit *AutoLog* vrijednost nase servis-a u true te ce se logirati samo osnove informacije poput uspijesno pokretanja ili zaustavljanja servisa itd.

Logove moozemo vidjeti unutar *Event Viewera* prema imenu logera kojeg definiramo pri inicijalizaciji EventLoger klase ili mozemo filtrirat Application logove prema imena naseg servisa

## Instalacija i pokretanje

Prvo moramo izbuildati projekt, poželjno sa *Release* konfiguracijom jer u nekim slučajevim *Debug* konfiguracija moze dovesti do problem tijekom pokretanja.

Nakon toga u VS-u otvorimo Developer Command Prompt te se pozicioniramo u terminalu unutar Release build foldera i pokrenemo naredbu:

**installutil *ime\_projekta*.exe**

Nakon toga u *Services Control Manager* ako je instalacija procesuirana uspješno nas servis trebao biti vidljiv

# Worker Service .NET Core