Servicii, Thread-uri şi Domenii de aplicaţie

Obiective

După completarea acestui laborator veți dobândi următoarele cunoștințe:

- Crearea de clase installer pentru instalarea unui serviciu
- Crearea unui setup project pentru rularea claselor installer
- Instalarea și dezinstalarea unui serviciu
- Scrierea de cod pentru thread-uri

Scenariu de laborator

Lucrați la o aplicație pentru monitorizarea fișierelor dintr-un computer. Computerul este folosit ca un server de fișiere, prin intermediul căruia diferite conturi pot partaja, crea, modifica și șterge fișiere. Clienții folosesc serverul pentru a încărca fișiere compresate; când un fișier este modificat serverul va deschide arhiva și va cataloga conținutul acestuia. Trebuie să scrieți codul necesar pentru a detecta momentul modificării fișierelor. Aplicația trebuie să dețină suficiente permisiuni pentru a accesa fișierele, de aceea soluția trebuie să fie un serviciu Windows, rulând pe un cont Windows, cu acces la sistemul de fișiere. Soluția returnează rezultate la una sau mai multe aplicații client, de aceea trebuie să fie o aplicație multithread.

➢ Exerciţiul 1- Crearea proiectului de instalare

În acest exercițiu veți adăuga un proiect **Windows Service** unei soluții existente, veți adăuga comunicări interprocese, și cod pentru serviciu, pentru a oferi o implementare simplă la comenzile client.

Taskul 1:

- Deschideți soluția oferită ca startup pentru laborator
- Observați cele două proiecte existente
  - **IPC**
  - **FileWatcherClient**
- Faceți click pe **File**, click **Add**, apoi click **New Project**
- Alegeți tipul proiectului **Windows Service**
- Numele proiectului va fi **FileWatcherService**
- Observați că clasa **Service1** este derivată din clasa **ServiceBase**

- Există două metode override **OnStart** și **OnStop**
- Observati metoda **Main** din fisierul **Program.cs**. Aceasta adauga o instanta a serviciului nostru la un vector de servicii pe care le porneste ulterior.

Taskul 2:

- Modificați proprietatea **ServiceName** la valoarea **FileWatcher** . Pentru a modifica proprietatea dati dublu-click pe **Service1.cs** si cautati proprietatea **ServiceName** in tabul **Properties**.
- Faceți click dreapta pe **FileWatcherService**, și apoi click pe **Add Reference**.
- Din tabul **Solution/Projects** selectați intrarea **IPC**.
    - Unitatea de asamblare este adăugată în folderul **References**.
- În clasa **Service1** adăugați suport pentru namespace-ul **IPC**
- Expandați nodul pentru proiectul **IPC**.
- Observați delegatul **ReceivedCommandDelegate**. Aceasta este apelat când este primită o comandă și este folosit pentru a transmite serviciului să trateze comanda.
    - Evenimentul **mreStop** este folosit pentru a informa toate thread-urile folosite să își înceteze activitatea.
    - Metoda **AcceptCallback** obține socket-ul activ folosind obiectul de stare.
    - Metoda **HandleConnection** are o buclă pentru a accepta în mod repetat comenzi de la client și pentru a le trata prin ridicarea evenimentului **ReceivedCommand**.
- Observați clasa **ClientSide**
    - Metoda **Connect** crează socketul și apoi se conectează
    - Metoda **Disconnect** deconectează sochetul
    - Metoda **Send** trimite o comandă la server și apoi așteaptă un reply.

Taskul 3:

- Adăugați un câmp clasei **Service1**, de tipul **ServerSide**.
- Instanțiați câmpul în constructorul clasei (Pasati ca parametru portul **5000**).
- În metoda **OnStart** apelați metoda **Listen**
- În metoda **OnStop** apelați metoda **StopListening**
- Adăugați o metodă handler pentru evenimentul **ReceiveCommand** al clasei **ServerSide,** folosind delegatul **ReceiveCommandDelegate**. Pentru a face acest lucru este nevoie de crearea unei noi metode in fisierul **Service1.cs**.
- In aceasta metoda implementați codul de extragere a comenzii și de apelare a metodelor handler pentru tratarea fiecărei comenzi
- Fiecare comandă are un nume urmat de un ID unic pentru fiecare client și apoi orice alți parametrii necesari comenzii.

    - Codul pentru această metodă este în fișierul **server_ReceivedCommand.txt**. Copiați conținutul acestui fișier în metoda **server_ReceivedCommand**.

Parametrii comenzii	Metodă handler	Descriere

START ID NumeFolder	OnStart	Transmite serviciului să verifice folderul pentru un anumit proces.
FINISH ID	OnFinish	Transmite serviciului să oprească verificarea folder-ului și să returneze orice rezultat.
RESET ID	OnReset	Transmite serviciului să returneze rezultatele, dare să continue să verifice folderul.
RESULTS ID	OnResults	Returnează rezultatele pentru folderele verificate de acest proces.

- Metoda **server_ReceivedCommand** extrage comanda și id-ul clientului din textul comenzii, apoi apelează metodele corespunzătoare de tratare a comenzii.
- Adăugați implementări pentru aceste metode(**ReceivedCommand_methods.txt**)
- Compilați proiectul și corectați eventualele erori.

➢ Exercițiul 2- Crearea proiectului de instalare

În acest exercițiu veți adăuga cod pentru instalarea serviciului. Pentru aceasta veți adăuga clase **Installer** proiectului de serviciu și un proiect de implementare soluției. Veți instala serviciul folosind o fereastră **Command Prompt** , rulând din contul de **Administrator**.

Taskul 1:

- Faceți dublu click pe **Service1.cs**, pentru a afișa componenta designer.
- Click dreapta pe aceasta, apoi click **Add Installer**.
- Wizard-ul va deschide o nouă fereastră designer, care conține următoarele componente:

Componentă	Descriere
serviceInstaller1	Installer pentru serviciu
serviceProcessInstaller1	Installer pentru proces

- S-a adaugat o componenta noua in cadrul proiectului cu numele **ProjectInstaller.cs,** componenta ce contine **serviceInstaller1** si **serviceProcessInstaller1** aflate in **ProjectInstaller.cs/ProjectInstaller.Designer.cs**.
- Click dreapta pe metoda **InitializeComponent** (din fisierul **ProjectInstaller.cs/ProjectInstaller**), apoi **Go to definition**
    - La sfârșitul metodei sunt adăugate în colecția **Installers** clasa **serviceInstaller1** și **serviceProcessInstaller1** prin intermediul metodei **AddRange**

- Pentru **serviceInstaller1** (obiectul poate fi accesat din designer-ul lui **ProjectInstaller.cs** – se ajunge acolo cu dublu-click pe acesta), modificați proprietățile acesteia folosind fereastra **Properties**

Proprietate	Valoare
DisplayName	File System Watcher
Description	Serviciul verifică foldere și raportează asupra fișierelor modificate

- Pentru **serviceProcessInstaller1**, modificați proprietatea **Account** la valoarea **LocalSystem**.
- Compilați și rezolvați eventualele erori.

Taskul 2:

- Pentru a putea folosi serviciul este nevoie de instalarea lui in sistem (stocarea de informatii in registre, etc). Acest lucru se face prin utilitarul **InstallUtil.exe** aflat in calea **„C:\Windows\Microsoft.NET\Framework\v4.0.30319”** (ultimul folder poate diferi in functie de versiunea de .NET)
- Deschideti o consola **cmd** ca administrator si navigati catre folderul de mai sus pentru a accesa executabilul.
- Rulati comanda **InstallUtil.exe „<cale_catre_FileWatcherService.exe>”** (FileWatcherService.exe se afla in „bin/Debug” al proiectului **FileWatcherService**. Comanda va instala serviciul.
- Verificati daca instalarea a avut loc prin cautarea acestuia in lista de servicii din **Task Manager**, tabul **Services**. Acesta ar trebui sa aiba starea **Stopped**.

➢ Exercițiul 3- Crearea unei aplicații client

În acest exercițiu veți implementa codul client pentru accesarea serviciului. Codul este parțial compilat și folosește clasa **ClientSide** pentru a oferi comunicare între procese folosind serviciul peste socket. Veți folosi aplicația client pentru a apela serviciul.

Taskul 1:

- Deschideți fișierul **MainForm.cs** din folderul **FileWatcherClient**. Principalele controale sunt descrise în tabelul următor:

Control	Descriere
Start	Trimite un mesaj de START serviciului, și un folder de verificat.
Stop	Transmite serviciului să oprească verificarea

	fișierelor.
Get Results	Returnează ultimele acțiuni realizate după ultimul Start sau Reset.
Reset	Șterge toate rezultatele stocate pentru client
List View	Sunt afișate toate rezultatele serviciului

- Uitați-vă peste restul codului clasei **ClientSide** din fisierul **IPC.cs**.
- Deschideti **MainForm.cs/MainForm.** Codul de interfață cu utilizatorul (**thread 0)** și thread-ul lucrător (**thread 1)**, accesează aceleași câmpuri (**cmd**, **watchFolder** și **id**). Deoarece **id** este inițializat o dată și apoi devine read-only, el poate fi accesat fără sincronizare. Pentru celelalte două câmpuri, asigurați sincronizare folosind cuvântul cheie ***lock*** sau clasa **Monitor**(hint: metoda **ListeningThread**, **btnStart_Click**, **btnStop_Click, btnResults_Click, btnReset_Click**). Aveti 2 optiuni: veti face lock pe obiectul curent (**this)** sau veti crea la nivel de clasa un obiect dummy pe care sa faceti lock.
- Compilați programul

Taskul 2:

- Porniți serviciul din fereastra **Command Prompt** deschisă anterior folosind comanda:

```
net start ”File System Watcher”
```

- Va fi afișat următorul text:

```
The File System Watcher service is starting.
The File System Watcher service was started succesfully.
```

- Setati proiectul **FileWatcherClient** ca startup project. În **Visual Studio** faceți click pe **Debug**, apoi pe **Start Debugging**, pentru a rula codul client.
- La deschiderea ferestrei client, realizați următoarele acțiuni:

Acțiune	Descriere
Click **Start**	Afișează fereastra de dialog Browse For Folder. Căutați folderul rădăcină E:. Click OK. A fost apelată metoda OnStart pe serviciu.
Click **Get Results**	A fost apelată metoda OnResults pe serviciu.
Click **Reset**	A fost apelată metoda OnReset pe serviciu.
Click **Stop**	A fost apelată metoda OnStop pe serviciu.

- Închideți fereastra.

Taskul 3:

- Instanța curentă de **Visual Studio** rulează pe contul de **Student**, iar pentru a face debugging trebuie să ruleze pe contul de **Administrator**. Deschideți **Visual Studio** folosind contul de **Administrator**.

- Deschideți soluția de laborator. În **Service1**.cs, duceți-vă la metoda **server_ReceivedCommand** și apoi faceți click pe prima linie de cod din interiorul acesteia
- Click **Debug**, apoi click **Toggle Breakpoint**.
- Click pe **Debug → Attach to Process →Show processes from all users → FileWatcherService.exe**
- În prima instanță de **Visual Studio** faceți click pe comanda **Start Without Debugging**. Apare fereastra client. Apăsați pe butonul de **Start** și apoi căutați folderul rădăcină E:. Click OK.
- Fereastra **WatcherClient** afișează **Start Work** în lista view control. După câteva secunde a doua instanță de **Visual Studio** apare și indică faptul că serviciul este în mod debugging și că s-a ajuns la breakpoint.
- La instanța care face debugging, pentru a detașa debugger-ul faceți click **Debug**, apoi click **Detach All**.
- Închideți fereastra care rulează pe contul de utilizator.
- Închideți fereastra **WatcherClient**.
- Opriți serviciul. Din fereastra **Command Prompt** tastați următorul cod:

```
net stop "File System Watcher"
```

- Dezinstalați serviciul (acest lucru este necesar pentru ca la urmatorul exercitiu va trebui sa modificam serviciul, prin urmare va fi nevoie de o noua instalare):

```
InstallUtil.exe /u „<cale_catre_FileWatcherService.exe>"
```

➢ Exercițiul 4 - Scrierea codului pentru serviciu

În acest exercițiu veți scrie cod pentru tratarea comenzilor de la un client. Serviciul poate gestiona apeluri de la mai mulți clienți, astfel va trebui să scrieți cod pentru sincronizare.

Taskul 1:

- Deschideți Service1.cs.
- Serviciul acceptă comenzi de la diferiți clienți. Fiecare cerere pornește un obiect **FileSystemWatcher**. Doi clienți pot cere ca același folder să fie verificat, astfel ne trebuie o clasă care să asigure că un singur obiect verifică un folder la un moment dat. Acest lucru este realizat de o clasă numită **IndexedWatcher**. Această clasă este furnizată în fișierul **IndexedWatcher.txt.**

- Copiați codul deasupra constructorului din **Service1**. Noua clasa va fi imbricata clasei **Service1.**
- Includeți namespace-urilor **System.IO** și **System.Threading**.
- Clasa **IndexedWatcher** trebuie sincronizată, de aceea schimbați codul pentru a restricționa lista **ProcessesWatching** la câte un thread la un moment dat.

- Folosiți câte un ***lock*** pentru codul care accesează această listă din metodele **AddProcess** și **RemoveProcess**.
- Adăugați in clasa **Service1** un câmp de tip **Dictionary** cu numele **data**, a cărui cheie este numărul de identificare al clientului, și ca valoare o listă de string-uri. Acest câmp va fi folosit pentru a menține o evidență client-folder.

Taskul 2:

- În constructor inițializați câmpurile watchers și data.
- Implementați metoda **OnStart (ATENTIE: cea care primeste 3 parametri)**:
  - Ștergeți linia de cod existentă
  - Adăugați codul din fișierul **OnStart.txt**
- Implementați metoda **OnStop (ATENTIE: cea care primeste 3 parametri)**:
  - După apelarea metodei **Report** adăugați codul din fișierul **OnFinish.txt**
- Implementați metoda **OnResults (ATENTIE: cea care primeste 3 parametri)**:
  - După apelarea metodei **Report** adăugați codul din fișierul **OnResults.txt**

- Implementați metoda **OnReset (ATENTIE: cea care primeste 3 parametri)**:
  - După apelarea metodei **Report** adăugați codul din fișierul **OnReset.txt**

- Metoda **OnStart** crează obiecte **IndexedWatcher** cu un delegat la metoda **FileEventHandler**. Codul este furnizat în fișierul **FileEventHandler.txt**. După metoda **OnFinish** copiați conținutul acestui fișier.
- Din moment ce ambele câmpuri **data** și **watchers** sunt accesate în această metodă, trebuie să folosiți un lock, precum în exemplul următor:

```
lock (watcher.ProcessesWatching)
lock ( data )
{
   foreach (int id in watcher.ProcessesWatching)
   {
         data[i].Add(str);
   }
}
```

Taskul 3:

- Tratați eventualele erori
- Dati **build** la solutie, dupa care reinstalati si porniti serviciul.
- Testati pe un folder (**atentie, numele folderului nu trebuie sa contina spatii)** crearea si stergerea unor fisiere sau directoare.
- Pentru a vedea modificarile aduse folderului trebuie sa se apese butonul „Get Results”.