***Сервисы: понятие и примеры служб, управление службами, алгоритм создания службы.***

***Разработайте программу, которая проверяет доступ к веб-адресу по протоколу HTTP и позволяет сохранить изображение по введенной пользователем ссылке на компьютер.***

**Службы Windows** — это исполняемые приложения, которые работают в фоновом режиме. Они не имеют графического интерфейса и могут быть запущены как при старте операционной системы, так и в любой другой момент работы пользователя.

Примеры служб:

автообновление Windows;

проверка электронной почты;

печать документов;

агент SQL Server;

сканирование и индексирование файлов и папок и др..

**Управление службами** осуществляется через диспетчер управления службами (SCM). Службы можно запускать, приостанавливать и перезапускать вручную.

**Алгоритм создания службы на C#:**

Создать новый проект в Visual Studio, выбрав тип Windows Service.

В окне редактора свойств настроить необходимые свойства, например: Autolog (события о запуске и останове службы должны автоматически записываться в журнал событий), CanPauseAndContinue, CanShutdown и CanStop (позволяют выполнять запросы на приостановку, возобновление, завершение и останов службы).

Открыть файл Program.cs и отредактировать метод Main для создания экземпляра службы.

Открыть файл Service.cs и реализовать методы OnStart() и OnStop(). Эти методы автоматически переопределяются при создании проекта.

**Сервис, или служба (в контексте Windows),** — это приложение, которое выполняется в фоновом режиме без пользовательского интерфейса и обычно предназначено для выполнения долгосрочных операций. Службы работают независимо от текущего пользователя, который вошел в систему, и могут запускаться автоматически при старте операционной системы. Они часто используются для выполнения задач, которые не требуют постоянного взаимодействия с пользователем, например:

Системные службы: Службы, которые управляют основными функциями операционной системы (например, служба печати, служба DNS-клиента, служба обновления Windows).

Службы веб-серверов: Сервисы, которые обрабатывают HTTP-запросы от клиентов (например, IIS, Apache, Nginx).

Службы баз данных: Сервисы, которые управляют базами данных (например, SQL Server, MySQL, PostgreSQL).

Службы резервного копирования: Сервисы, которые автоматически выполняют резервное копирование данных.

Службы мониторинга: Сервисы, которые отслеживают состояние системы и сообщают об ошибках.

Службы электронной почты: Сервисы, которые отправляют и принимают электронные письма.

Пользовательские службы: Приложения, разработанные пользователем для выполнения специфических задач, которые требуют фонового режима.

**Управление службами в Windows можно осуществлять различными способами:**

Инструмент “Службы” (services.msc):

Графический инструмент для просмотра списка установленных служб, их статуса, типа запуска (автоматический, ручной, отключен), и для управления ими (запуск, остановка, перезапуск, изменение типа запуска).

Можно запустить, набрав services.msc в окне поиска или команде “Выполнить”.

Командная строка (cmd.exe):

Можно использовать команды net start, net stop и sc (Service Controller) для управления службами.

net start <имя\_службы> - запустить службу.

net stop <имя\_службы> - остановить службу.

sc query <имя\_службы> - посмотреть информацию о службе.

sc config <имя\_службы> start= <тип\_запуска> - изменить тип запуска.

start= auto - автоматический.

start= demand - ручной.

start= disabled - отключен.

PowerShell:

Можно использовать командлеты Get-Service, Start-Service, Stop-Service, Restart-Service для управления службами.

Get-Service <имя\_службы> - посмотреть информацию о службе.

Start-Service <имя\_службы> - запустить службу.

Stop-Service <имя\_службы> - остановить службу.

Restart-Service <имя\_службы> - перезапустить службу.

Управление службами через программный код:

Службами можно управлять через программный код, используя API Windows (например, System.ServiceProcess в .NET).

**Шаги создания службы на C#:**

Создание проекта:

Откройте Visual Studio.

Выберите “Создать проект”.

Выберите шаблон “Служба Windows” (Windows Service) и укажите имя проекта.

Visual Studio автоматически создаст проект с необходимыми файлами.

Изменение логики службы (Service1.cs по умолчанию):

В созданном проекте вы увидите класс, унаследованный от System.ServiceProcess.ServiceBase.

Переопределите методы OnStart, OnStop, OnPause, OnContinue и OnShutdown, в которых будет реализована логика вашей службы.

OnStart: Вызывается при запуске службы. В этом методе обычно размещают код инициализации службы.

OnStop: Вызывается при остановке службы. В этом методе обычно размещают код очистки ресурсов.

OnPause: Вызывается при приостановке службы.

OnContinue: Вызывается при возобновлении службы.

OnShutdown: Вызывается при выключении системы.

Добавление таймера (опционально):

Для выполнения периодических задач можно использовать System.Timers.Timer или System.Threading.Timer.

**Установка службы:**

Для установки службы необходимо использовать утилиту InstallUtil.exe (входит в состав .NET Framework SDK).

Запустите Developer Command Prompt от имени администратора.

Перейдите в каталог, где находится скомпилированный EXE файл службы.

Выполните команду InstallUtil.exe <имя\_файла\_службы>.exe

**Запуск службы**:

Запустите службу через инструмент “Службы” (services.msc), командную строку (net start <имя\_службы>) или PowerShell (Start-Service <имя\_службы>).

**Удаление службы:**

Для удаления службы нужно использовать InstallUtil.exe с опцией /u.

Выполните команду InstallUtil.exe /u <имя\_файла\_службы>.exe

**Пример кода службы на C# (Service1.cs):**

using System;

using System.Diagnostics;

using System.IO;

using System.ServiceProcess;

using System.Timers;

namespace MyService

{

public partial class Service1 : ServiceBase

{

private Timer timer;

private int eventId = 1;

private string logFilePath = "C:\\MyServiceLog.txt";

public Service1()

{

InitializeComponent();

}

protected override void OnStart(string[] args)

{

WriteLog("Service started");

timer = new Timer(60000); // интервал 60 секунд

timer.Elapsed += new ElapsedEventHandler(OnTimer);

timer.Start();

}

private void OnTimer(object sender, ElapsedEventArgs e)

{

WriteLog("Timer event: " + eventId++);

}

protected override void OnStop()

{

WriteLog("Service stopped");

timer.Stop();

}

protected override void OnPause()

{

WriteLog("Service paused");

timer.Stop();

}

protected override void OnContinue()

{

WriteLog("Service continued");

timer.Start();

}

private void WriteLog(string message)

{

try

{

File.AppendAllText(logFilePath, DateTime.Now.ToString() + " : " + message + Environment.NewLine);

}

catch(Exception ex)

{

EventLog.WriteEntry("Service error logging",ex.ToString(),EventLogEntryType.Error,0);

}

}

}

}