<http://wladm.narod.ru/C_Sharp/services.html>

Службы Windows (Windows NT services) - процессы, обладающие унифицированным интерфейсом для взаимодействия с операционными системами Windows серии NT. Службы делятся на два типа - службы Win32, взаимодействующие с операционной системой посредством диспетчера управления службами (Service Control Manager - SCM), и драйвера, работающие по протоколу драйвера устройства (далее речь идет о службах Win32). Службы работают в фоновом режиме, и их работа скрыта от пользователя. В силу этого они идеально подходят для реализации серверных процессов в архитектуре клиент-сервер, сетевых служб, программ мониторинга и спорадически выполняемых программ.

Для создания службы в VS .NET необходимо выполнить:

* Создать проект решения службы с помощью специального шаблона Visual C# Projects - Windows Service.
* Написать функциональный код и код событий OnStart и OnStop.
* Создать установщики для службы (ProjectInstaller - для установки процесса и ServiceInstaller - для службы или служб проекта).
* Создать исполняемый файл.
* Создать проект установки.
* Установить и активизировать службу.

### *Параграф 1. Создание проекта службы*

В меню File VS .NET выбераем New и Project. В диалоговом окне New Project Project Types выбираем Visual C# Projects, Template - Windows Service, задаем Name, например, MyFirstServicee, Location - определяем директорию, где будем формировать проект.

После нажатия кнопки OK проект решения Service будет создан. Cозданные файлы проекта несколько отличны от тех, что были в Visual Studio 2003. Их имена (см. Рис.1. и директорию проекта):  
- Имя\_Проекта.sln,  
- Имя\_Проекта.csproj,  
- Servece1.Designer.cs,  
- Service1.cs,  
- Service1.resx.

Файл AssemblyInfo.cs "переехал" в папку Properties.

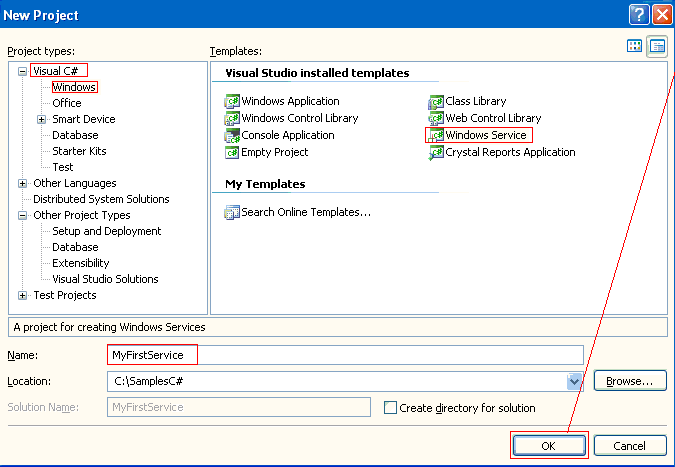
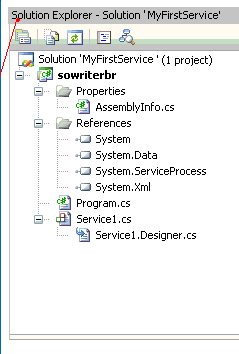


Рис.1. Создание проекта решения службы

На данном этапе нас интересует файл с именем Service1.cs. Service - имя класса проекта, наследника класса System.ServiceProcess.ServiceBase.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics;

using System.ServiceProcess;

using System.Text;

namespace MyFirstService

{

public class Service1 : ServiceBase

{

....

Класс содержит конструктор класса, функцию main и функции методов OnStart и OnStop. Методы OnPause (), и OnContinue () также могут быть включены в проект (их код может быть дописан вручную, но методы не являются обязательными).

//Конструктор класса

public Service1()

{

InitializeComponent();

}

static void Main()

{

}

protected override void OnStart(string[] args)

{

//Код, выполняемый при старте процесса

}

protected override void OnStop(string[] args)

{

//Код, выполняемый при остановке процесса

}

Код инициализации компонент в Visual Studio 2005 перемещен в файл Servece1.Designer.cs, как и метод Dispose:

namespace MyFirstService

{

partial class Service1

{

private System.ComponentModel.IContainer components = null;

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Component Designer generated code

private void InitializeComponent()

{

components = new System.ComponentModel.Container();

this.ServiceName = "Service1";

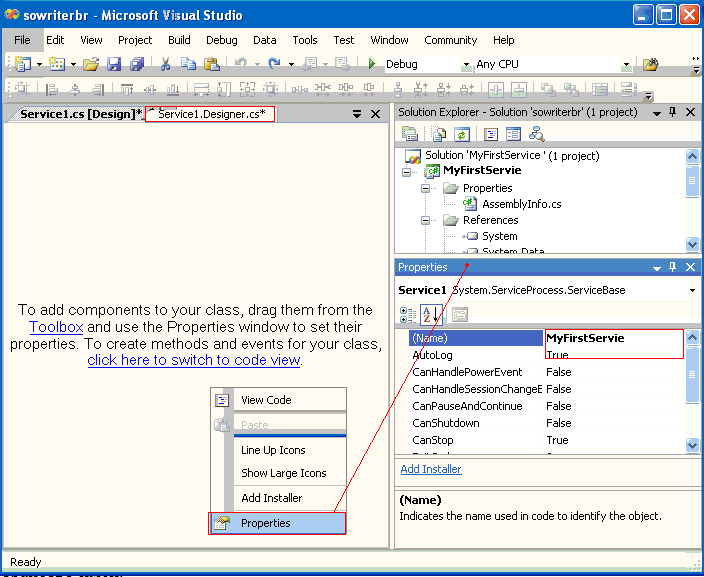
}

#endregion

}

}

В контекстном меню дизайнера формы (вкладка проекта Service1.cs [Design], правый клик мышки) выбираем пункт Properties (если вкладка не видна - дважды кликнем мышкой по имени файла Servece1.cs в Solutation Explorer). Устанавливаем свойства(Рис.2.):  
- ServiceName (имя проекта) - MyFirstService,   
- AutoLog - True.

Рис.2. Установка свойств сервиса

Здесь можно изменить и отображаемое имя сервиса (свойство ServiceName).

На этом проект решения сервиса создан.

*Параграф 2. Написание функционального кода сервиса*

Для того, чтобы убедиться в работоспособности нашего сервиса, возложим на него задачу писать в некоторый лог файл какую либо информацию при старте и останове. Для этого:

* Добавим пространство имен:

using System.IO;

* Объявим в классе Service1 StreamWriter:

private StreamWriter file;

* Коды обработчиков событий запишем следующим образом:

protected override void OnStart(string[] args)

{

//Файл по умолчанию будет создан в "C:\Winnt\System32\"

file = new StreamWriter(new FileStream("MyFirstService.log",

System.IO.FileMode.Append ));

this.file.WriteLine("MyFirstService стартовал");

this.file.Flush();

}

protected override void OnStop()

{

this.file.WriteLine("MyFirstService остановлен");

this.file.Flush();

this.file.Close();

}

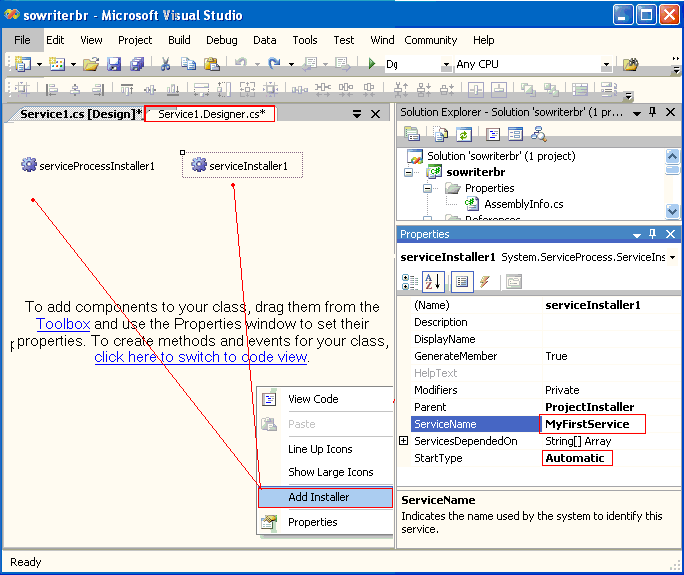
Позже мы добавим нашему сервису другую функциональность.

*Параграф 3. Создание установщиков для службы*

На вкладке Service1.cs [Design] вызываем вновь контекстное меню и пункт Add Instaler. В проект добавляется класс компонента, содержащий два установщика:  
- ProjectInstaller - для установки службы и  
- ServiceInstaller - для установки связанного со службой процесса.

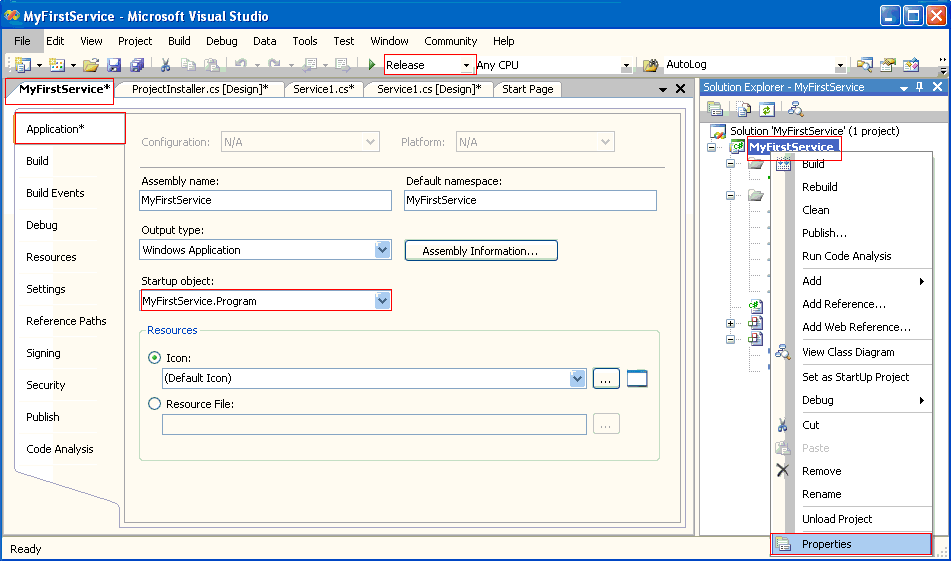
Для ServiceInstaller1 отображаем его свойства (контекстное меню Properties) и задаем:  
- ServiceName - **MyFirstService**,  
- ServerType - **Automati**c.

Для ServiceProcessInstaller1 устанавливаем значение **Account** в **LocalSystem** (Рис.3.).

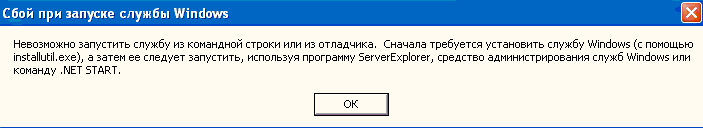
Рис.3. Создание установщика для службы

*Параграф 4. Создание исполняемого файла для службы*

В Solution Explorer выбираем пункт Properties для узла MyFirstServices. В диалоговом окне выбираем вкладку Application. Для пункта Startup object выбираем в выпадающем списке MyFirstService.Programm (Рис.4.).

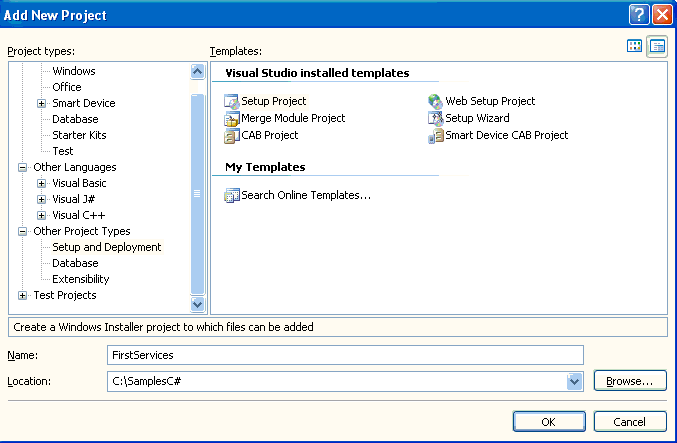
Рис.4. Установка свойств для программы установки

В контекстном меню для узла MyFirstServices выбираем пункт Build. После этого, проект построен и в директории проекта obj\debug или obj\release будет сформирован файл MyFirstService.exe. Однако, попытка его запуска приведет к выдаче сообщения (Рис.5.):

Рис.5. Необходимость установки сервиса

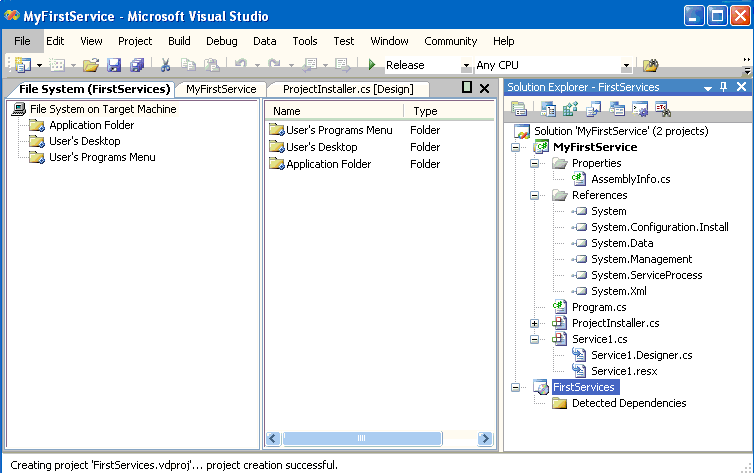
*Параграф 5. Создание проекта установки для службы*

В меню File выбираем пункт Add Project и New Project. В диалоговом окне Add New Project в окне Project Types выбираем вкладку Other Project Types ветвь Setup and Deployment, в окне Templates - Setup Project, задаем Name и Location - оба параметра непринципиальны(Рис.6.).

Рис.6. Добавление проекта решения установки службы

Проект установки добавлен в решение и мы его видим в Solutation Explorer.

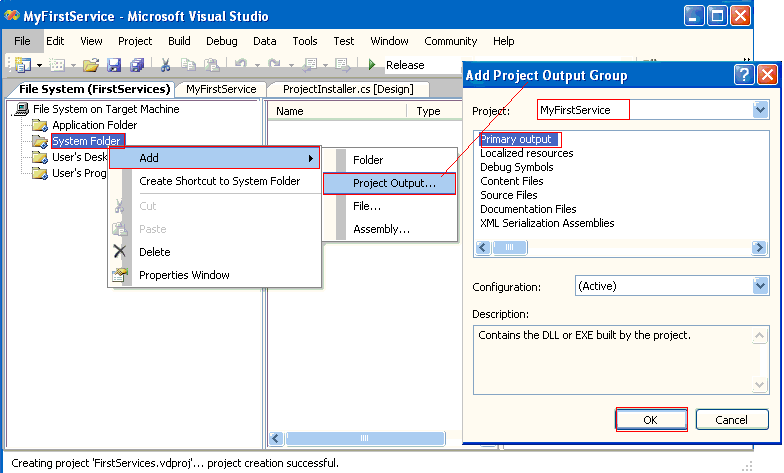
Кроме того на вкладке File System (FirstServices) мы можем видеть сборку файлов для установки "File System on Target Machine" (Рис.7.).

Рис.7. Создание проекта решения установки службы

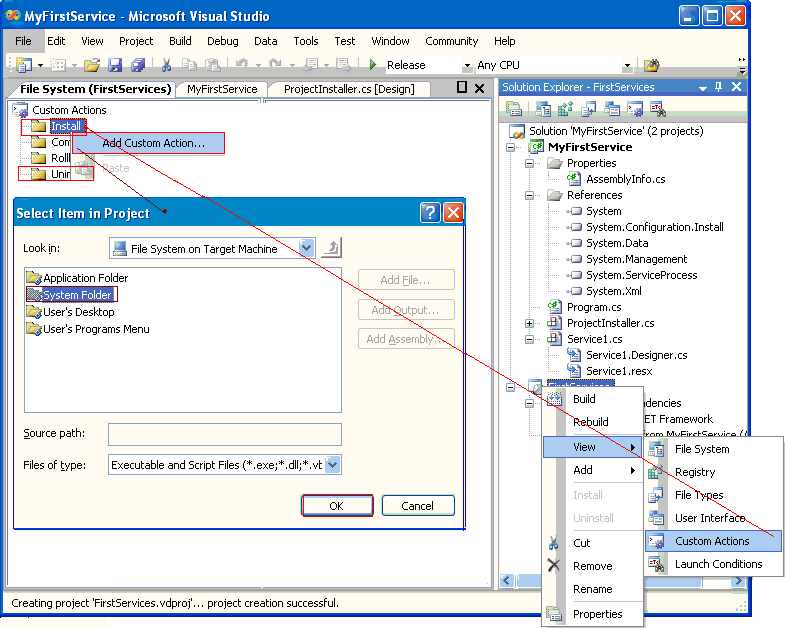
Следующее, что нужно сделать - это определить для установки директорию для проекта службы Windows.

Выбираем узел File System on Target Machine (если по какой либо причине мы не видим дерево папок, то можно его отобразить через контекстное меню узла FirstServices, пункт View/File System), в его контекстном меню выбираем Add Special Folder и System Folder. Имя System Folder появилось в ветви узла File System on Target Machine.

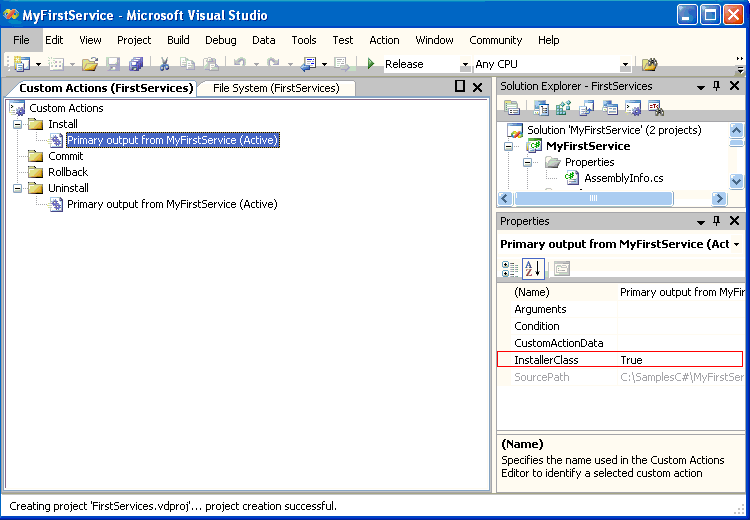
В контекстном меню System Folder выбираем Add, Project Output\Primary Output (Рис.8.).

Рис.8. Определение директории для службы

Выбираем пункт View для узла FirstServices пункт Custom Action, слева в окне откроются по умолчанию четыре возможных действия для инсталлятора: Install, Commit, Rollback и Uninstall. Для Install и Uninstall через их контекстное меню выполняем пункт Add Custom Action и указываем System Folder (Рис.9.).

Рис.9. Определение действий инсталлятора

Практически, мы определили не только действие, но и объект этого действия - Primary output from MyFirstService (Active). В данные узлы пользовательских действий (Install и Uninstall) будет добавлен основной выходной файл проекта. В свойствах можно изменить для файла CustomInstaller, значение true сообщает инсталлятору, что используется класс инсталлятора (Рис.10.).

Рис.10. Определение действий инсталлятора

Теперь можно построить проект инсталлятора, вызвав для узла FirstServices в Solutation Explorer пункт Build.

*Параграф 6. Установка, активизация и удаление службы*

Для установки службы в Solutation Explorer выбираем проект установки (FirstServices) и в контекстном меню пункт Install. В результате, будет запущен Setup Wizard. На втором шаге его работы зададим системную папку Windows - C:\WINDOWS\system32\. После окончания работы Setup Wizard служба будет установлена.

Чтобы запустить и остановить службу откроем диспетчер управления службами (в Windows XP - Start/Programm/Administrative Tools/Services). Имя нашей службы MyFirstService будет отображено в окне Services, и можно обычным образом запустить наш сервис через контекстное меню пункт Start (кроме того, служба автоматически будет запускаться при перегрузке компьютера).

Наш сервис (служба) создал файл Service1.log в "C:\windows\System32\ и выполнил в него запись о своем старте. При останове сервиса запись будет выполнена вновь.

Службу можно устанавливать и удалять, используя файлы setup.exe и FirstServices.msi, которые после построения установщика находятся в каталоге, где был создан проект инсталлятора В данном примере в C:\.....\FirstService\Debug\ и, кроме того, этих двух файлов достаточно для установки службы из любой директории любого компьютера.

На данном этапе мы выполнили все требуемые шаги и проверили нашу службу в работе. Далее, можно выполнить построение проекта в режиме Release (меню Build, Configuration Manager, Release). setup.exe и FirstServices.msi, после построения установщика будут находятся в каталоге C:\.....\FirstServices\bin\Release). Далее возможно наполнять функциональность службы по своему усмотрению.

Удаления сервиса с компьютера выполняется как и обычного приложения Windows - (Settings/Control Panel/Add on Remove Programs).

*Параграф 7. Пример использования службы, для периодического выполнения некоторых действий*

Для задания простого механизма опроса можно воспользоваться компонентом System.Timers.Timer. В методе OnStart можно задать параметры компонента. Таймер будет выполнять в коде периодически некоторые действия (например, запись в файл текущего времени, как это показано ниже). В код лишь добавлен компонент Timer и обработчик события, выполняющий некоторые действия по истечении заданного свойством timer1.Interval времени.

namespace MyFirstService

{

public class Service1 : ServiceBase

{

private StreamWriter file;

//Определяем таймер

private System.Timers.Timer timer1;

......

protected override void OnStart(string[] args)

{

file = new StreamWriter(new FileStream("Service1.log",

System.IO.FileMode.Append ));

this.file.WriteLine("MyFirstService стартовал");

this.file.Flush();

//Создаем таймер и выставляем его параметры

this.timer1 = new System.Timers.Timer();

this.timer1.Enabled = true;

//Интервал 10000мс - 10с.

this.timer1.Interval = 100000;

this.timer1.Elapsed +=

new System.Timers.ElapsedEventHandler(this.timer1\_Elapsed);

this.timer1.AutoReset=true;

this.timer1.Start();

}

protected override void OnStop()

{

this.timer1.Stop();

this.file.WriteLine("MyFirstService остановлен");

this.file.Flush();

this.file.Close();

}

private void

timer1\_Elapsed(object sender, System.Timers.ElapsedEventArgs e)

{

//Записываем время в файл или делаем все, что хотим

this.file.WriteLine(DateTime.Now.ToString("dd.MM.yyyy HH:mm:ss"));

this.file.Flush();

}

Содержимое файла Service1.log:

MyFirstService стартовал

21.06.2005 15:01:59

21.06.2005 15:02:09

21.06.2005 15:02:19

21.06.2005 15:02:29

MyFirstService остановлен

*Молчанов Владислав 21.06.2005г.*

*Адаптировано к VS 2005 16.04.2008г.*

*Проверено в VS 2010 8.09.2011г.*