

Версия для  
печати  
(включая  
изображения)



Отправить  
ссылку



Поместить  
в блог



Добавить  
комментарий



Написать  
автору



Подписаться  
на рассылки

### Конфигурации флэш-массивов Dell 20

ИИ выпустила новые  
и флэш-массивов Dell Storage  
ого класса. В рамках своей  
ансформации рынка систем ...

ешение «ИнфоТеКС» для  
правления контентом  
нфоТеКС» и «Национальный  
эжки и разработки»  
совместное решение  
Инфооборот»,  
нное для ...

### ла сети LTE в 76 регионах

ГС подвела некоторые итоги  
звертыванию своих сетей в  
за прошедший год оператор  
сети в 63 регионах ...

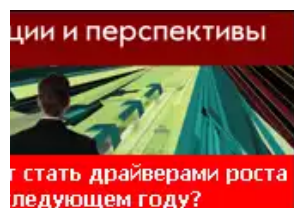
### Me-Room в ЦОД DataLine

ю компании DataLine, в ее  
ентре OST 3 на ул. Боровая  
туги Meet-Me-Room (MMR). Эта  
ляет оперативно ...

### ые адаптеры D-Link

Link представила 10-  
етевые адаптеры DXE-810T и  
шины PCI Express,  
нные для применения в

### вост



Раздел: Программное обеспечение

№5 (57), май 2003

## Управление внешними программами в среде .NET Framework

12.05.2003



Наращение  
конкурентной борьбы

Андрей Колесов

Одна из обычных задач, возникающих при разработке ПО, - запуск и отслеживание состояния внешних программ. Традиционно при программировании в Windows для этого приходилось использовать средства Win API. Функции, появившиеся в технологии .NET, существенно упрощают решение данной задачи. Рассмотрим эти новые возможности на примере VB.NET.

### Знакомство с классом Process

В классическом Visual Basic запуск внешних приложений выполнялся с помощью функции Shell, например, так:

```
ReturnID = Shell ("calc.exe", vbNormalFocus)
```

Эта функция в .NET была несколько улучшена по сравнению с VB 6.0, и ею по-прежнему можно пользоваться, однако ее возможности весьма ограничены, прежде всего из-за того, что вызываемое приложение запускается в асинхронном режиме.

Вместо этого для работы с внешними программами в .NET лучше использовать класс Process, находящийся в пространстве имен System.Diagnostics. В простейшем случае запуск внешней программы будет выполняться с помощью метода Start; в данном примере для обработки указанного файла будет запускаться текстовый редактор (обычно это NotePad, установленный по умолчанию):

```
System.Diagnostics.Process.Start _  
("c:\MYPATH\MYFILE.TXT")
```

Метод Start, в свою очередь, возвращает объект Process. С его помощью можно получить ссылку на запущенный процесс, например, чтобы узнать его имя:

```
Dim myProcess As Process = _  
Process.Start("c:\MYPATH\MYFILE.TXT")  
MsgBox(myProcess.ProcessName)
```

Читайте в  
выпуске с  
«Бестселлеры

№04/2014

Новые т

на рынок

### Темы

- Перспективы технологий
- Информационная безопасность
- Серверы
- HPC (High Performance Computing)
- Центры обработки данных
- Технологии данных
- Инфраструктура технологий

чем преимущество all-flash массивов?  
инято считать, что при мерении производительности Д основной акцент делается с ввода-вывода (IOPS): чем их выше ...

И N – не просто эволюционный иг в развитии систем хранения трошлом году многие оизводители СХД заявили о ппуске программно их платформ (SDN). Это не оционный шаг, а очередной

обходимое условие победы в гменте серверов для SMB – рокий портфель продуктов инок серверов для среднего и лого бизнеса дробится на зких ниш со своими особыми производительности, ности и ...

ения

е читаемые

унифицированные ации Orange лвные жесткие диски Toshiba 6 Тбайт сервис SAP HANA – первый оссии серии HP Stream – в продаже libaba открывает офис в

годом, дорогие читатели! эское портфолио Cisco для лющего Интернета рсии корпоративных решений 32 юнные решения Avaya дули памяти AMD для

енные решения

стема управления ИТ-фраструктурой предприятия – джер»

фраструктурные решения от rapon Software: преимущества изнеса

vell ZENworks Configuration inagement

.Web Security Space

ЕЛО» версии 12.2.1: новые зможности для венного взаимодействия

шения

е курсы

aptec Certified Storage ofessional

ратно за парту с Adaptec by /С

осы

ка на рассылки

Для управления параметрами запускаемого процесса можно воспользоваться объектом ProcessStartInfo из того же пространства имен:

```
Dim psInfo As New _
    System.Diagnostics.ProcessStartInfo _
    ("c:\mypath\myfile.txt")
' устанавливаем стиль открываемого окна
psInfo.WindowStyle = _
    System.Diagnostics.ProcessWindowStyle.Normal
Dim myProcess As Process = _
    System.Diagnostics.Process.Start(psInfo)
```

Объект ProcessStartInfo можно также получить с помощью свойства Process.Start:

```
Dim myProcess As _
    System.Diagnostics.Process = _
    New System.Diagnostics.Process()
myProcess.StartInfo.FileName = _
    "c:\mypath\myfile.txt"
myProcess.StartInfo.WindowStyle = _
    System.Diagnostics.ProcessWindowStyle.Normal
myProcess.Start()
```

Предварительную установку параметров запускаемого процесса (имя файла, стиль окна и т. п.) можно выполнять и в режиме разработки (Design Time) через компонент Process, который следует добавить к форме из раздела Components панели Toolbar. В этом случае все параметры свойства StartInfo можно будет вводить в окне Properties (рис. 1).

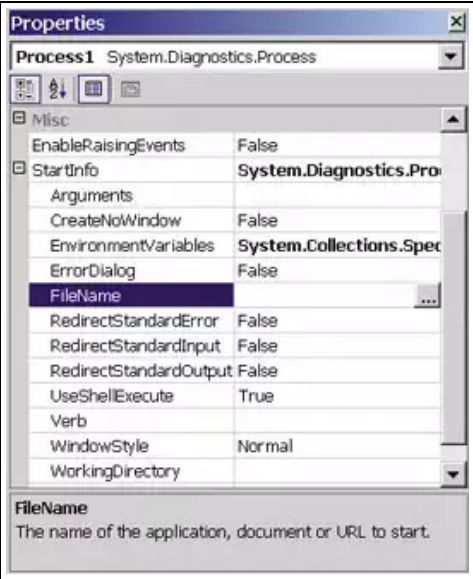


Рис. 1. Управлять параметрами объекта Process можно в среде разработки.

Запуск процесса и ожидание его завершения

Самый простой способ ожидания завершения запущенного процесса - использовать метод Process.WaitForExit (однако нужно иметь в виду, что, когда вы применяете его из Windows Form, форма перестает реагировать на некоторые системные события, например, Close):

```
' создание нового процесса
Dim myProcess As Process = _
    System.Diagnostics.Process.Start _
    ("c:\mypath\myfile.txt")
' Ожидание его завершения
myProcess.WaitForExit()
' вывод результатов
MessageBox.Show _
    ("Notepad был закрыт в: " & _
    myProcess.ExitTime & "." & _
    System.Environment.NewLine & _
    "Код завершения: " & _
    myProcess.ExitCode)
' закрытие процесса
myProcess.Close()
```

Здесь нужно обратить внимание на два момента. Во-первых, хотя запущенный процесс уже завершен, можно получить информацию о нем, например, узнать время его окончания или код завершения. Во-вторых, после завершения запущенного процесса все же нужно выполнить операцию его закрытия Process.Close, чтобы освободить память, отведенную под объект Process.

Метод WaitForExit можно использовать для ожидания завершения запущенного процесса в течение заданного интервала времени, а метод Kill - для его аварийного завершения:

```
' Ожидание в течение 5 с
myProcess.WaitForExit(5000)
' Если процесс не завершился, то мы
' аварийно завершаем его
```

- Периферии оборудова
- Телеком-ре
- Мобильные
- ERP
- Документо
- Управление процессами
- Бизнес-ана
- Интеграции технологий
- Разработка приложений
- Интеграция приложений
- Управление инфраструктурой
- Технологии виртуализа

Меропри

27 – 30 января  
г. | Москва  
Выставка «Бухучет и аудит-2

5 – 6 февраля  
г. | Москва  
«Инфофорум-2

Другие мер

Форумы

Беспроводное данных: новые возможности | Seagate  
Рома, 20.08.20

Спорт и ИТ  
Гость PFIZER, 15:41:35

Платёжные си  
Гость Киви, 21 11:52:39

Документообс  
Гость Элар, 20 14:08:29

ИТ в Медици  
Гость, 14.12.20

Пресс-ре

«1С:Управлени автотранспорт помогает

«1С-Рарус:Копитания» - для

Cisco расширя образовательные инициативы в

Другие пресс-г

## Express

ы новых статей, последние  
ти и т.п.

Подписаться

е подпиской

```
If Not myProcess.HasExited Then  
    myProcess.Kill()  
    ' все же ждем его завершения  
    myProcess.WaitForExit()  
End If
```

Обратите внимание: после выполнения метода Kill мы опять ожидаем завершения процесса. Это нужно сделать, если мы хотим затем получить информацию о времени завершения и код завершения, - иначе при обращении к свойствам ExitTime или ExitCode будет выдана программная ошибка, так как запущенный процесс еще не успеет закончиться.

## Запуск скрытого процесса

Довольно часто разработчику вообще не нужно, чтобы внешний процесс отражался в окне на экране монитора. Типичный случай - выполнение какой-то операции в сеансе MS-DOS, например, получение списка файлов заданного каталога. Приведенный ниже код выполняет подобную операцию без создания окна для запущенного процесса. Обратите внимание на содержание командной строки: Windows XP распознает "&&" как разделитель команд, позволяя записывать в одну строку сразу несколько команд.

```
Dim myProcess As Process = New Process()  
' имя выходного файла  
Dim outfile As String = _  
    Application.StartupPath & _  
    "\dirOutput.txt"  
  
' адрес системного каталога  
Dim sysFolder As String = _  
    System.Environment.GetFolderPath _  
    (Environment.SpecialFolder.System)  
  
' Имя запускаемой программы  
' и строка аргументов  
myProcess.StartInfo.FileName = "cmd.exe"  
myProcess.StartInfo.Arguments = _  
    "C cd " & sysFolder & _  
    " && dir *.com >> " & Chr(34) & _  
    outfile & Chr(34) & " && exit"  
  
' запуск процесса в скрытом окне  
myProcess.StartInfo.WindowStyle = _  
    ProcessWindowStyle.Hidden  
myProcess.StartInfo.CreateNoWindow = True  
myProcess.Start()  
  
myProcess.WaitForExit() 'ожидание
```

## Определение момента завершения процесса

Как мы видели в предыдущем примере, метод WaitForExit блокирует все события приложения, которое инициировало внешний процесс. Чтобы дать возможность работать главной программе, используется конструкция, которая в цикле проверяет состояние внешнего процесса и тут же вызывает метод Application.DoEvents, обеспечивающий обработку других событий данного приложения:

```
Do While Not myProcess.HasExited  
    Application.DoEvents  
Loop
```

Кстати, эффект, получаемый в данном случае при опросе свойства HasExit, в VB 6.0 можно было получить, обратившись к функции Win32 API GetModuleUsage.

Однако с точки зрения минимизации загрузки процессора более эффективный по сравнению с предыдущим примером вариант - инициализация события Exited класса Process. При этом нужно установить свойство Process.EnableRaisingEvents равным True (по умолчанию оно равно False) и создать манипулятор события, включая процедуру обработки события:

```
Private Sub Button4_Click _  
    (ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) _  
    Handles Button4.Click  
  
    Dim myProcess As Process = New Process()  
    myProcess.StartInfo.FileName = _  
        "c:\mypath\myfile.txt"  
    ' разрешаем процессу  
    ' инициализировать событие  
    myProcess.EnableRaisingEvents = True  
    ' Добавить манипулятор события Exited  
    AddHandler myProcess.Exited, _  
        AddressOf Me.ProcessExited  
    ' запуск процесса  
    myProcess.Start()  
End Sub  
  
Friend Sub ProcessExited _  
    (ByVal sender As Object, _  
    ByVal e As System.EventArgs)  
    ' процедура обработки события  
    ' (завершение процесса)  
    Dim myProcess As Process =
```

```

Dim myProcess As Process = _
    DirectCast(sender, Process)
MessageBox.Show _
    ("Процесс завершился в" & _
    myProcess.ExitTime & _
    System.Environment.NewLine & _
    "Код завершения: " & _
    myProcess.ExitCode)
myProcess.Close()
End Sub

```

Здесь нужно обратить внимание на одну потенциальную опасность: если вызванный процесс зависнет, то с приложением тоже могут возникнуть проблемы. Чтобы избежать этого, можно включить контроль времени ожидания по таймеру.

### Обмен информацией с внешним процессом

Иногда возможностей передачи информации в вызываемый процесс через простую командную строку оказывается явно недостаточно. Бывает также, что получение от него результирующих данных через файл, как это сделано в предыдущем примере, не представляется оптимальным вариантом. Порой для обмена данными требуются механизмы прямого взаимодействия основного и вызываемого приложений.

Для вызываемых программ, которые поддерживают механизмы StdIn, StdOut и StdErr (например, консольных приложений), можно использовать объекты StreamWriter и StreamReader для записи и чтения данных. Чтобы это сделать, нужно установить свойства RedirectStandardInput, RedirectStandardOutput и RedirectStandardError объекта ProcessStartInfo равными True. Затем, после запуска внешнего процесса, нужно использовать свойства StandardInput, StandardOutput и StandardError объекта Process для привязки потока ввода-вывода к объектам StreamReader и StreamWriter.

Еще одно замечание: по умолчанию среда .NET Framework использует функцию Win32 ShellExecute для взаимодействия с внешним процессом, но когда вы переопределяете потоки ввода-вывода, перед запуском приложения нужно установить свойство ProcessStartInfo.UseShellExecute равным False.

В приведенном ниже примере создается невидимое окно, формируется список файлов заданного каталога, а результаты выводятся в окне MessageBox (рис. 2)



Рис. 2. Информация, полученная из вызванного процесса через объект StdOut.

```

' обмен данными с внешним процессом через
' функции StdIn, StdOut и StdErr
Dim s As String
Dim myProcess As Process = New Process()
' описание и запуск процесса
myProcess.StartInfo.FileName = "cmd.exe"
myProcess.StartInfo.UseShellExecute = False
myProcess.StartInfo.CreateNoWindow = True
myProcess.StartInfo.RedirectStandardInput = True
myProcess.StartInfo.RedirectStandardOutput = True
myProcess.StartInfo.RedirectStandardError = True
myProcess.Start()

' описание объектов передачи данных
Dim sIn As System.IO.StreamWriter = _
    myProcess.StandardInput
sIn.AutoFlush = True
Dim sOut As System.IO.StreamReader = _
    myProcess.StandardOutput
Dim sErr As System.IO.StreamReader = _
    myProcess.StandardError
' передача входных данных
sIn.Write("dir c:\drv\*. *" & _
    System.Environment.NewLine)
sIn.Write("exit" & _
    System.Environment.NewLine)
' получаем результат выполнения команды DIR

```

```

s = sOut.ReadToEnd()
If Not myProcess.HasExited Then
    myProcess.Kill()
End If

MessageBox.Show("Окно команды 'dir' " & _
    "было закрыто в: " & _
    myProcess.ExitTime & "." & _
    System.Environment.NewLine & _
    "Код завершения: " & _
    myProcess.ExitCode)

sIn.Close()
sOut.Close()
sErr.Close()
myProcess.Close()

' смотрим результат работы процесса DIR
MessageBox.Show(s)

```

Если вызываемое приложение не использует StdIn, можно применить метод SendKeys для передачи данных, вводимой с клавиатуры. Например, следующий код вызывает NotePad и вводит в него некоторый текст:

```

Dim myProcess As Process = New Process()
myProcess.StartInfo.FileName = "notepad"
myProcess.StartInfo.WindowStyle = _
    ProcessWindowStyle.Normal
myProcess.Start()

' Ждем 1 с, чтобы NotePad был готов
' к вводу данных
myProcess.WaitForInputIdle(1000)
if myProcess.Responding Then
    System.Windows.Forms.SendKeys.SendWait( _
        "Этот текст был введен " & _
        "с помощью метода " & _
        "System.Windows.Forms.SendKeys.")
Else
    myProcess.Kill()
End If

```

Метод SendKeys позволяет передавать коды любых клавиш, включая Alt, Ctrl и Shift. Таким образом можно передавать комбинации клавиш для выполнения стандартных команд, например, загрузки или сохранения файлов, управления командами меню и т. п. Но нужно помнить, что этот метод посылает код только в активное окно приложения, и если нужное окно потеряет фокус, могут возникнуть проблемы. Именно поэтому мы использовали метод Process.WaitForInputIdle, чтобы проверить, готово ли приложение к получению информации. Для NotePad времени ожидания в 1 с вполне достаточно, но для других приложений его, возможно, придется увеличить.

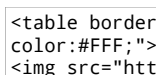
\*\*\*

Итак, хотя функция Shell по-прежнему работает в .NET Framework, класс System.Diagnostics.Process предоставляет гораздо больше возможностей для взаимодействия с внешними процессами. Переадресуя потоки StdIn, StdOut и StdErr, можно наладить двусторонний обмен данными с приложением, а применяя метод SendKeys, можно вводить информацию (в том числе управляющие команды меню) в программы, которые не используют StdIn.

## Другие статьи из раздела

- ▶ Программные инструменты криминалистов: можно ли нарушить закон в онлайн и остаться непойманным?
- ▶ Национальные особенности рынка ЕСМ/СЭД в России
- ▶ Корпоративная мультиплатформенность бизнес-приложений
- ▶ Что нового в новой версии виртуальной машины Oracle?
- ▶ Облачная стратегия Microsoft 2011

[Поместить в блог](#)

<a href="http://www.bytemag.ru" target="_blank">  </a>
<a href="http://www.bytemag.ru/articles/detail.php?ID=8942" style="font: 14px Arial; color: #000; text-decoration: none; font-weight: bold;" target="_blank">             Управление внешними программами в среде .NET Framework         </a>

Скопировать код

Предпросмотр

[Комментарии к статье](#)

-----\*



Ваше имя\*\*

Ваш комментарий\*

Защита от автоматических сообщений\*



Введите символы на картинке

Добавить комментарий

\* - Поля, обязательные для заполнения.



#### Рекламные ссылки

[BYTEmag.ru приглашает к сотрудничеству авторов статей по ИТ-тематике.](#)

[Твердотельные массивы меняют рынок СХД](#)



#### Фоторепортажи



**Chloride**  
**Демонстрация Chloride Trinergy**  
Впервые в России компания Chloride Rus провела демонстрацию системы бесперебойного электропитания Chloride Trinergy®, а также ИБП Chloride 80-NET™, NXС и NX для своих партнеров и заказчиков.



**NEC Нева Коммуникационные Системы**  
**Завершена реорганизация двух дочерних предприятий NEC Corporation в России**  
С 1 декабря 2010 года Генеральным директором ЗАО «NEC Нева Коммуникационные Системы» назначен Раймонд Армес, занимавший ранее пост Президента Shyam ...



**компания «Гротек»**  
**С 17 по 19 ноября 2010 в Москве, в КВЦ «Сокольники», состоялась VII Международная выставка InfoSecurity Russia. StorageExpo. Documation'2010.**  
Новейшие решения защиты информации, хранения данных и документооборота и защиты персональных данных представили 104 организации. 4 019 руководителей ...

[Другие фоторепортажи](#)



#### Видеоролики



**Adaptec by PMC**  
**RAID-контроллеры Adaptec Series 5Z с безбатарейной защитой кэша**

Опытные сетевые администраторы знают, что задействование в работе кэш-памяти RAID-контроллера дает серьезные преимущества в производительности ...



**Chloride**  
**Трехфазный ИБП Chloride от 200 до 1200 кВт: Trinergy**

Trinergy — новое решение на рынке ИБП, впервые с динамическим режимом работы, масштабируемостью до 9.6 МВт и КПД до 99%. Уникальное сочетание ...



**Embarcadero**  
**Возможности Delphi 2010**

Посмотрите обзор среды разработки Delphi и оцените, как можно использовать данную технологию для быстрой разработки Windows-приложений. В это обзор ...

[Другие видеоролики](#)



