Topshelf分享

官网：<http://docs.topshelf-project.com/en/latest/index.html>

## 几个重要概念说明

1. **什么是Topshelf？**

Topshelf是为.NET平台设计的一个Windows服务框架。

1. **为什么我要用Topshelf？**

Topshelf让创建、测试、调试和安装Windows服务变得极为简单。开发者只需要关注服务要处理的逻辑，不必关心复杂的服务类细节以及如何使用其核心的安装类将其安装到平台上以及如何附加调试模块对服务进行调试等。这样就大大节省了开发者的精力。

1. **Topshelf怎么做到这些？**

当开发者使用Topshelf，他只需要创建一个控制台程序，然后创建一个带公共的Start和Stop方法的类，用Topshelf API简单配置一下，就完成了一个Windows的创建，并且可以对其进行调试（F5）和安装（通过命令行）。

1. **如果我的服务需要定制配置项怎么办？**

大多数的Windows服务配置项Topshelf都包括了，比如启动项的自动、自动（延迟启动）、手动、禁用，本地服务、网络服务、启动依赖项（包括SQL Server、MSMQ等）等，具体见官网说明。

1. **它对我本身的服务有什么影响？**

Topshelf是一个很小（约200K）的程序集并且无任何依赖项，所以它很容易跟任何服务结合使用。

1. **命令行参数如何？**

Topshelf有一个扩展的命令行集合，允许服务通过特定的命令行参数来注册参数。

1. **Topshelf只是针对Windows的吗？**

Topshelf本身是针对.NET平台设计的，但它在Mono上也可以用，所以它也可以被部署在Linux平台上。

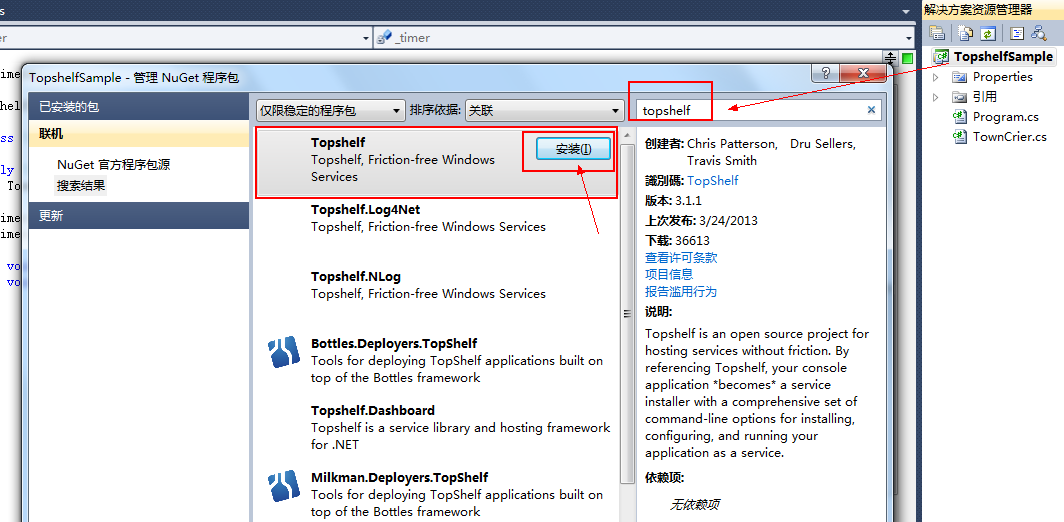
## Topshelf安装说明

1. **运行环境说明**

Topshelf是一个.Net框架所以它需要一个.Net运行环境（.NET 3.5 Service Pack 1 或 .NET 4.0），所以一般需要在Windows操作系统上运行。目前Windows 7、Windows Server 2003和Windows Server 2008RC2都没有问题。

1. **安装**

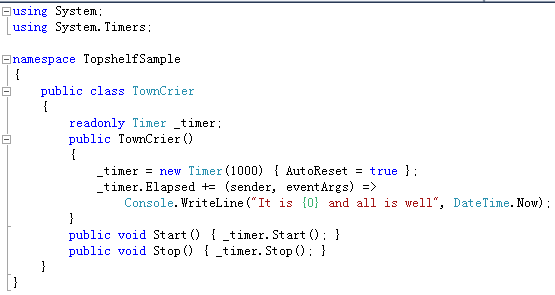
开发者可以使用Nuget安装Topshelf，也可以直接下载其程序集然后引用。下图演示的是用Nuget安装Topshelf



## Topshelf配置说明

### 代码展示

<http://docs.topshelf-project.com/en/latest/configuration/quickstart.html>





1: 这里通过HostFactory.Run安装这个host，通过其lambda表达式的x来进行各种host级别的参数配置。

2：这里我们告诉Topshelf这里有一个‘TownCrier’服务，lambda表达式的s用于设置各种服务级别的配置。

3：这里告诉Topshelf如何给这个服务构建一个实例，这里我们只是简单的将实例new出来，实际使用中通常会通过IoC来注入一个实例，代码类似‘container.GetInstance<TownCrier>()’.

4：告诉Topshelf启动这个服务时做什么。

5：告诉Topshelf如何停止这个服务时做什么。

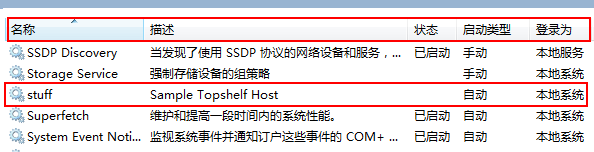
6：以什么身份启动该服务，这里选择的是登录为：本地系统local system。

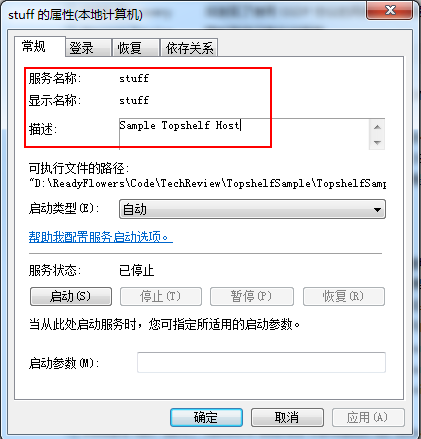
7：设置该服务的描述。

8：设置该服务的显示名称。

9：设置该服务自身的名字，即服务名称。

对于6、7、8、9的设置在下图可一目了然：





### 各项配置

下面列举一些简单常用的配置项，更多配置请参考官网：

<http://docs.topshelf-project.com/en/latest/configuration/config_api.html>

1. **Topshelf Configuration**
2. Service Name 服务名称

设置用于向服务控制器注册的服务的基本名称，该项可选，未设置的情况下默认是Program.cs文件的命名空间；

HostFactory.New(x =>

{

x.SetServiceName("MyService");

});

建议服务名称不要包含空格和空白符；

一个操作系统内每个服务必须有一个唯一区分的名字，所以如果想在一个系统内装同一个服务的多个实例的话，需要在用命令行注册服务时使用不同的实例名。

1. Service Description 服务描述

对服务的描述，也是可选项，默认与服务名称相同；

HostFactory.New(x =>

{

x.SetDescription("My First Topshelf Service");

});

1. Display Name 显示名称

它是指该服务在Windows服务控制台上的显示名称，也是可选项，默认与服务名称相同。

HostFactory.New(x =>

{

x.SetDisplayName("MyService");

});

1. Instance Name 实例名称

服务的实例名称，与基本的服务名称相连，用$隔开；该项可选，且默认下不会设置，除非明确声明了：

HostFactory.New(x =>

{

x.SetInstanceName("MyService");

});

实例名一般是在命令行参数里面进行设置的，但这里也允许进行声明。

1. **Service Start Modes 服务启动模式**

HostFactory.New(x =>

{

x.StartAutomatically(); // Start the service automatically 自动

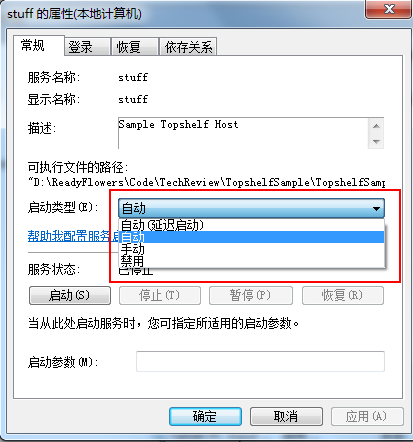
x.StartAutomaticallyDelayed(); // Automatic (Delayed) -- only available on .NET 4.0 or later

x.StartManually(); // Start the service manually

x.Disabled(); // install the service as disabled

});

对应的各种启动项如下图所示：



1. **Service Identity运行服务的身份**
2. 以给定的用户名和密码运行，它也可在命令行里面配置

HostFactory.New(x =>

{

x.RunAs("username", "password");

});

1. 当安装该服务时，提示输入用户名密码后再启动服务

HostFactory.New(x =>

{

x.RunAsPrompt();

});

1. 使用内置的NETWORK\_SERVICE帐户运行

HostFactory.New(x =>

{

x.RunAsNetworkService();

});

1. 以本地帐户身份运行

HostFactory.New(x =>

{

x.RunAsLocalSystem();

});

1. 以本地服务帐户运行

HostFactory.New(x =>

{

x.RunAsLocalService();

});

1. **Custom Install Actions 定制安装行为**

此类配置允许在服务安装或卸载时运行给定的代码

1. Before Install Actions 服务安装前运行给定代码

HostFactory.New(x =>

{

x.BeforeInstall(() => { ... });

});

1. After Install Actions 服务安装后运行给定代码

HostFactory.New(x =>

{

x. AfterInstall (() => { ... });

});

1. Before Install Actions 服务卸载前运行给定代码

HostFactory.New(x =>

{

x. BeforeUninstall (() => { ... });

});

1. Before Install Actions 服务卸载后运行给定代码

HostFactory.New(x =>

{

x. AfterUninstall (() => { ... });

});

1. **Service Dependencies 服务依赖项**

设置依赖项后，只有当依赖项启动后该服务才会启动

HostFactory.New(x =>

{

x.DependsOn("SomeOtherService");

});

Topshelf内置了一些服务依赖项设置方法，包括：

HostFactory.New(x =>

{

x.DependsOnMsmq(); // Microsoft Message Queueing

x.DependsOnMsSql(); // Microsoft SQL Server

x.DependsOnEventLog(); // Windows Event Log

x.DependsOnIis(); // Internet Information Server

});

### 与logging结合

默认情况下，Topshelf使用TraceSource来记录日志，它是.Net框架自带的，所以不会引入任何依赖项。但是，很多应用会使用更高级的日志记录类库，比如log4net 或 NLog等。所以，Topshelf也提供了一个扩展的日志记录接口用于支持这些高级日志记录类库。具体使用请参考官网：

<http://docs.topshelf-project.com/en/latest/configuration/logging.html>

## Topshelf命令行参照

<http://docs.topshelf-project.com/en/latest/overview/commandline.html>

服务一旦被创建，就可以在命令行里对该服务进行各种配置，通过输入myService.exe help可以获得命令行的使用说明：

service.exe **[verb]** [-option:value] [-switch]

**run**运行服务 (default) **help**, –help 显示帮助项

**install** 安装服务

|  |  |
| --- | --- |
| -u*sername* | 运行服务的用户名 |
| -p*assword* | 运行服务的指定用户名的密码 |
| -i*nstance* | 指定某一个实例（如果该服务注册了多个实例的话） |
| --autostart | 服务自启动 (默认选择) |
| --disabled | 禁用该服务 |
| --manual | 手动启动 |
| --delayed | 延迟启动(delayed) |
| --localservice | 使用本地local service account运行服务 |
| --networkservice | |
|  | 以network service权限运行服务 |
| --interactive | 安装该服务时提示输入用户名密码再运行服务 |
| --sudo | 如果运行在Vista/W7/2008下的话弹出UAC |
| -s*ervicename* | 安装服务时的服务名称 |
| -d*escription* | 安装服务时的服务描述 |
| -d*isplayname* | 安装服务时的显示名称 |

**start** 如果服务未运行的话启动该服务

|  |  |
| --- | --- |
| -i*nstance* | 要启动的服务实例 |

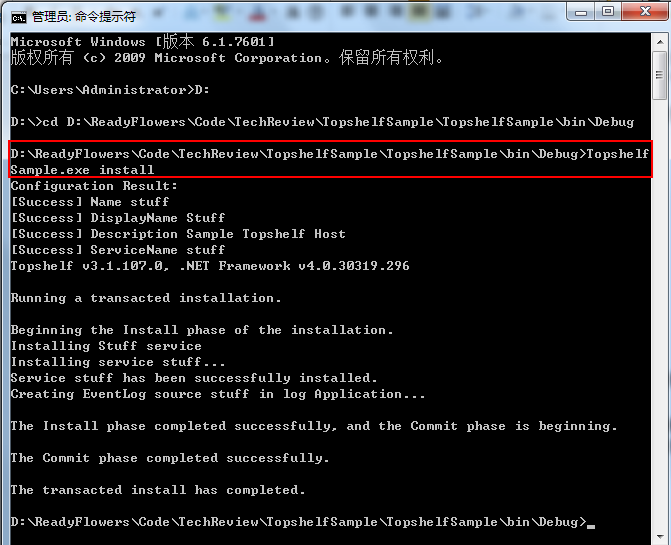
**stop** 如果服务在运行的话停止它

|  |  |
| --- | --- |
| -i*nstance* | 要停止的服务实例的名称 |

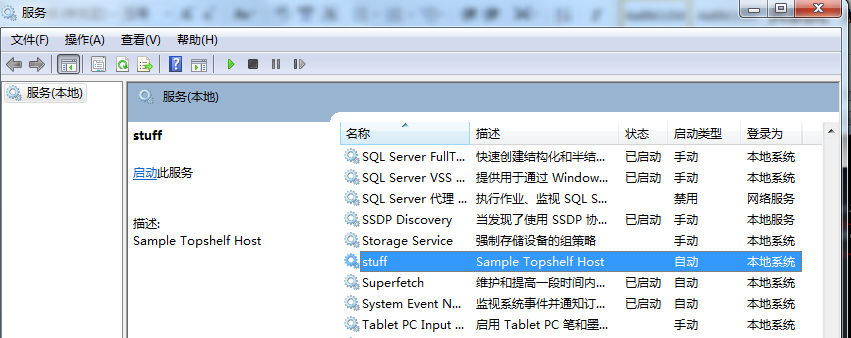
**uninstall** 卸载该服务

|  |  |
| --- | --- |
| -i*nstance* | 要卸载的服务实例的名称 |
| --sudo | 如果运行在Vista/W7/2008下的话弹出UAC |

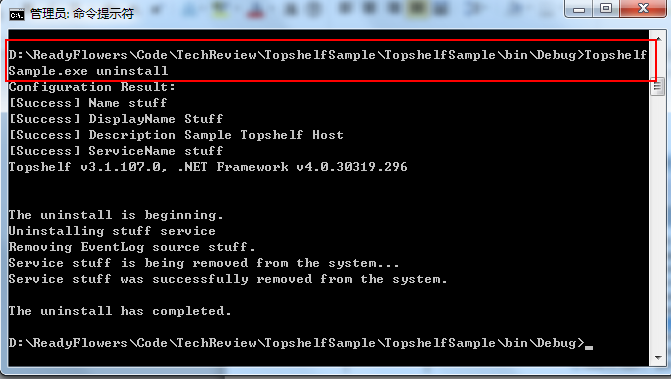
如果一切都是默认的话，常用的两个命令行就是安装服务和卸载服务，比如安装Demo里面创建的服务，应输入：TopshelfSample.exe install，然后服务就被安装到系统了，界面如下：



然后在Windows服务控制台就能看到刚刚安装的服务：



若要卸载该服务，则输入TopshelfSample.exe uninstall，界面如下：



## 附件Demo

见：TopshelfSample.7z