**Service Name**

指定服务的基名称，因为它在服务控制管理器注册。此设置是可选的且默认情况下使用 Program.cs 文件的命名空间 (好吧，基本上，调用程序集类型命名空间) 。

HostFactory.New(x =>

{

x.SetServiceName("MyService");

});

建议服务名称不包含空格或其他空白字符。  
系统上的每个服务必须有一个唯一的名称。如果你需要运行同一个服务的多个实例，注册服务时请考虑使用实例名的命令行选项。

**Service Description**

在服务控制管理器中指定该服务的描述。这是可选的默认为该服务名称。

HostFactory.New(x =>

{

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

});

**Display Name**

在服务控制管理器中指定服务的显示名称。这是可选的默认为该服务名称。

HostFactory.New(x =>

{

x.SetDisplayName("MyService");

});

**Instance Name**

指定服务的实例名，这由基础服务名称和用 $ 分隔（待研究，实例名的格式，没有明白是什么格式）。这是可选的并且只添加的如果指定。

HostFactory.New(x =>

{

x.SetInstanceName("MyService");

});

此选项通常设置使用命令行参数，但这里允许出于完整性的考虑。

**Service Configuration**

The service can be configured in multiple ways, each with different goals. For services that can handle a dependency on Topshelf, the

ServiceControl

interface provides a lot of value for implementing the service control methods. Additionally, a zero-dependency solution is also available when lambda methods can be used to call methods in the service class.  
  
服务可以用多种方式配置，每种方式都有不同的目标。【可以处理 Topshelf 依赖的服务，】ServiceControl 接口提供了大量有价值用于实现服务控制方法（翻译的有点勉强）。此外，当 lambda 方法用于调用服务类中的方法时，0-依赖解决方案是可用的。

**Simple Service**

To configure a simple service, the easiest configuration method is available.  
  
若要配置一个简单的服务，最简单的配置方法是可用的。

HostFactory.New(x =>

{

x.Service<MyService>();

});

// Service implements the ServiceControl methods directly and has a default constructor

class MyService : ServiceControl

{}

If the service does not have a default constructor, the constructor can be specified, allowing the service to be created by the application, such as when a container needs to be used.  
  
如果服务不包含一个默认的构造函数，允许应用程序的服务指定构造函数，例如当一个容器需要被使用。

HostFactory.New(x =>

{

x.Service<MyService>(() => ObjectFactory.GetInstance<MyService>());

});

// Service implements the ServiceControl methods directly and has a default constructor

class MyService : ServiceControl

{

public MyService(SomeDependency dependency)

{}

}

If the service needs access to the HostSettings during construction, they are also available as an overload.  
  
如果服务在构造函数中需要读取HostSetting，他们作为过载也是有效的（意思就是可以通过构造函数的参数传递过去）

HostFactory.New(x =>

{

x.Service<MyService>(hostSettings => new MyService(hostSettings));

});

// Service implements the ServiceControl methods directly and has a default constructor

class MyService : ServiceControl

{

public MyService(HostSettings settings)

{}

}

**Custom Service 自定义服务**

To configure a completely custom service, such as one that has no dependencies on Topshelf, the following configuration is available.  
  
若要配置一个完整的自定义服务，例如，并不依赖于 Topshelf，以下配置是可用的。

HostFactory.New(x =>

{

x.Service<MyService>(sc =>

{

sc.ConstructUsing(() => new MyService());

// the start and stop methods for the service

sc.WhenStarted(s => s.Start());

sc.WhenStopped(s => s.Stop());

// optional pause/continue methods if used

sc.WhenPaused(s => s.Pause());

sc.WhenContinued(s => s.Continue());

// optional, when shutdown is supported

sc.WhenShutdown(s => s.Shutdown());

});

});

Each of the WhenXxx methods can also take an argument of the

HostControl

interface, which can be used to request the service be stopped, request additional start/stop time, etc.  
  
每个WhenXXX方法可以有一个HostControl接口类型的参数，它用于请求停止该服务，请求额外的起止时间等参数。

HostFactory.New(x =>

{

x.Service<MyService>(sc =>

{

sc.WhenStarted((s, hostControl) => s.Start(hostControl));

}

}

The

HostControl

interface can be retained and used as the service is running to Stop the service.  
  
HostControl 接口可以保留和用作该服务正在运行停止服务。

**Service Start Modes**

There are multiple service start modes, each of which can be specified by the configuration. This option is only used if the service is being installed.  
  
有多个服务启动模式，每一种可以由配置指定。这个选项只能用在服务被安装（启动）的时候。

HostFactory.New(x =>

{

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

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

x.StartManually(); // Start the service manually 手动模式

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

});

**Service Recovery**

Topshelf also exposes the options need to configure the service recovery options as well.  
  
Topshelf 还公开选择需要配置的服务恢复选项。

HostFactory.New(x =>

{

x.EnableServiceRecovery(r =>

{

//you can have up to three of these

r.RestartComputer(5, "message");

r.RestartService(0);

//the last one will act for all subsequent failures

r.RunProgram(7, "ping google.com");

//should this be true for crashed or non-zero exits

r.OnCrashOnly();

//number of days until the error count resets

r.SetResetPeriod(1);

});

});

**Service Identity 服务身份**

Services can be configured to run as a number of different identities, using the configuration option that is most appropriate.  
  
服务可以被配置为以不同的身份运行，使用以下选项配置最合适。（如果要以不同的身份运行服务，请使用以下方式配置）

HostFactory.New(x =>

{

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

});

Runs the service using the specified username and password. This can also be configured using the command-line.  
  
使用指定的用户名和密码运行服务。也可以再命令行中配置。

HostFactory.New(x =>

{

x.RunAsPrompt();

});

When the service is installed, the installer will prompt for the username/password combination used to launch the service.  
  
当安装了服务时，安装程序将提示输入用户名/密码组合用于启动该服务。

HostFactory.New(x =>

{

x.RunAsNetworkService();

});

Runs the service using the NETWORK\_SERVICE built-in account.  
  
使用NETWORK\_SERVICE 内置帐户运行服务。

HostFactory.New(x =>

{

x.RunAsLocalSystem();

});

Runs the service using the local system account.

HostFactory.New(x =>

{

x.RunAsLocalService();

});

Runs the service using the local service account.

**Custom Install Actions 自定义安装动作**

These settings allow user-specified code to be executed during the service install/uninstall process.  
  
在服务启动/卸载的过程中，允许执行用户指定代码。（即启动/卸载过程中允许用户执行自定义代码）

**Before Install Actions 安装前**

Topshelf allows actions to be specified that are executed before the service is installed. Note that this action is only executed if the service is being installed.

HostFactory.New(x =>

{

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

});

**After Install Actions 安装后**

Topshelf allows actions to be specified that are executed after the service is installed. Note that this action is only executed if the service is being installed.

HostFactory.New(x =>

{

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

});

**Before Uninstall Actions 卸载前**

Topshelf allows actions to be specified that are executed before the service is uninstalled. Note that this action is only executed if the service is being uninstalled.

HostFactory.New(x =>

{

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

});

**After Uninstall Actions 卸载后**

Topshelf allows actions to be specified that are executed after the service is uninstalled. Note that this action is only executed if the service is being uninstalled.

HostFactory.New(x =>

{

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

});

**Service Dependencies 服务依赖**

Service dependencies can be specified such that the service does not start until the dependent services are started. This is managed by the windows services control manager, and not by Topshelf itself.  
  
可以指定服务的依存关系，一个服务直到它依赖的服务启动之后才会运行。服务依赖由 windows 服务控制管理器管理，而不是Topshelf 本身。

HostFactory.New(x =>

{

x.DependsOn("SomeOtherService");

});

There are a number of built-in extension methods for well-known services, including:  
  
有一些内置的服务依赖扩展方法为知名的服务提供，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

});

**Advanced Settings 提升设置**

**EnablePauseAndContinue**

Specifies that the service supports pause and continue, allowing the services control manager to pass pause and continue commands to the service.  
  
指定服务支持暂停和继续，允许服务控制管理器向服务传送暂停和继续命令。

HostFactory.New(x =>

{

x.EnablePauseAndContinue();

});

**EnableShutdown 快速关闭服务**

Specifies that the service supports the shutdown service command, allowing the services control manager to quickly shutdown the service.

HostFactory.New(x =>

{

x.EnableShutdown();

});

**Service Recovery 服务恢复**

To configure the service recovery options, a configurator is available to specify one or more service recovery actions. The recovery options are only used when installing the service, and are set once the service has been successfully installed.  
  
若要配置的服务恢复选项，配置器可用来指定一个或多个服务恢复操作。恢复选项只用于在安装服务的过程中，当服务被成功安装时且只设置一次，注意，恢复操作是按指定的顺序执行的。

HostFactory.New(x =>

{

x.EnableServiceRecovery(rc =>

{

rc.RestartService(1); // restart the service after 1 minute

rc.RestartSystem(1, "System is restarting!"); // restart the system after 1 minute

rc.RunProgram(1, "notepad.exe"); // run a program

rc.SetResetPeriod(1); // set the reset interval to one day

})

});