Erstellen sicherer ASP.NET-Anwendungen

Authentifizierung, Autorisierung und sichere Kommunikation

Auf der Orientierungsseite finden Sie einen Ausgangspunkt und eine vollständige Übersicht zum *Erstellen sicherer ASP.NET-Anwendungen*.

Zusammenfassung

Für Objekte, die unter Verwendung der Remoteinfrastruktur von.NET aufgerufen werden, können ASP.NET, benutzerdefinierte ausführbare Dateien oder Windows-Dienste als Host fungieren. In dieser Vorgehensweise wird erläutert, wie ein Windows-Dienst als Host für ein Remoteobjekt verwendet wird und wie das Remoteobjekt von einer ASP.NET-Anwendung aus aufgerufen wird.

Vorgehensweise: Verwenden eines Windows-Dienstes als Host für ein Remoteobjekt

Nachstehend wird erläutert, wie ein Windows-Dienst als Host für ein Remoteobjekt verwendet wird und wie das Remoteobjekt von einer ASP.NET-Anwendung aus aufgerufen wird.

Hinweise

- Für Remoteobjekte (d. h. .NET-Objekte, auf die von Remotestandorten unter Verwendung der .NET Remoting-Technologie zugegriffen wird) können Windows-Dienste und benutzerdefinierte ausführbare Dateien ebenso wie ASP.NET als Host fungieren.
- Clients kommunizieren über den TCP-Kanal mit Remoteobjekten, für die benutzerdefinierte ausführbare Dateien oder Windows-Dienste als Host fungieren.
- Mit Remoteobjekten, für die ASP.NET als Host fungiert, kommunizieren Clients über den HTTP-Kanal.
- Wenn Sicherheit oberstes Gebot ist, sollte ASP.NET als Objekthost fungieren und der HTTP-Kanal verwendet werden. Auf diese Weise profitieren Sie von den zugrunde liegenden Sicherheitsfunktionen von ASP.NET und IIS.
  Informationen über das Hosten eines Remoteobjekts in ASP.NET (mit IIS) finden Sie in der Microsoft Knowledge Base im Artikel Q312107, "HOW TO: Host a Remote Object in IIS" (US).
- Wenn es in erster Linie um die Leistungsfähigkeit geht, sollte ein Windows-Dienst als Objekthost fungieren und der TCP-Kanal verwendet werden. Diese Option bietet jedoch keine integrierten Sicherheitsfunktionen.

Anforderungen

Im Folgenden finden Sie eine Liste der empfohlenen Hardware und Software und eine Beschreibung der Netzwerkinfrastruktur, der Fähigkeiten und Kenntnisse sowie der Service Packs, die Sie benötigen.

- Microsoft® Windows® 2000 als Betriebssystem
- Microsoft Visual Studio® .NET als Entwicklungssystem

Die hierin erläuterten Verfahren setzen zudem Kenntnisse des Entwicklungstools Microsoft Visual C#™ voraus.

Zusammenfassung

Diese Vorgehensweise enthält folgende Verfahren:

1. Erstellen der **RemoteObject**-Klasse
2. Erstellen einer Hostanwendung auf Basis eines Windows-Dienstes
3. Erstellen eines Windows-Kontos zum Ausführen des Dienstes
4. Installieren des Windows-Dienstes
5. Erstellen einer Test-Clientanwendung

1. Erstellen der RemoteObject-Klasse

In diesem Verfahren wird eine einfache **RemoteObject**-Klasse erstellt. Diese Klasse stellt eine einzige Methode mit Namen **Add** bereit, die zwei Zahlen addiert und das Ergebnis zurückgibt.

▶ **So erstellen Sie die RemoteObject-Klasse**

1. Starten Sie Visual Studio .NET, und erstellen Sie in Visual C# ein neues Klassenbibliotheksprojekt mit Namen **RemoteObject**.
2. Benennen Sie **Class1.cs** im Projektmappen-Explorer in **Calculator.cs** um.
3. Benennen Sie in **Calculator.cs** die Klasse **Class1** in **Calculator** um, und weisen Sie auch dem Standardkonstruktor den entsprechenden Namen zu.
4. Leiten Sie die **Calculator**-Klasse von **MarshalByRefObject** ab, um die Klasse remotefähig zu machen.

```
public class Calculator : MarshalByRefObject
```

5. Fügen Sie der **Calculator**-Klasse die folgende Methode vom Typ **public** hinzu.

```
public int Add( int operand1, int operand2 )
{
  return operand1 + operand2;
}
```

6. Klicken Sie im Menü **Erstellen** auf **Projektmappe erstellen**.

2. Erstellen einer Hostanwendung auf Basis eines Windows-Dienstes

Mit diesem Verfahren wird eine Windows-Dienstanwendung erstellt, die als Host für das Remoteobjekt verwendet wird. Wenn der Dienst gestartet wird, konfiguriert dieser den TCP-Remotingkanal so, dass eingehende Clientanforderungen abgefragt werden.

Hinweis: Bei diesem Verfahren werden eine **Installer**-Klasse und das Befehlszeilendienstprogramm **Installutil.exe** zum Installieren des Windows-Dienstes eingesetzt. Zum Deinstallieren des Dienstes führen Sie **Installutil.exe** mit der Option **/u** aus. Alternativ können Sie auch ein Setup- und Weitergabeprojekt zur Unterstützung bei der Installation und Deinstallation des Windows-Dienstes verwenden.

▶ **So erstellen Sie eine Hostanwendung auf Basis eines Windows-Dienstes**

1. Fügen Sie dem aktuellen Projekt mit Visual C# ein neues Windows-Dienstprojekt mit Namen **RemotingHost** hinzu.
2. Benennen Sie **Service1.cs** im Projektmappen-Explorer in **RemotingHost.cs** um.
3. Benennen Sie in **RemotingHost.cs** die Klasse **Service1** in **HostService** um, und weisen Sie auch dem Standardkonstruktor den entsprechenden Namen zu.
4. Fügen Sie am Anfang der Datei unterhalb der vorhandenen **using**-Anweisungen die folgende **using**-Anweisung hinzu.

```
using System.Runtime.Remoting;
```

5. Suchen Sie die **Main**-Methode, und ersetzen Sie die vorhandene Codezeile, mit der die **ServicesToRun**-Variable initialisiert wird, durch die folgende Zeile.

```
ServicesToRun = new System.ServiceProcess.ServiceBase[] {
                                            new HostService() };
```

6. Suchen Sie die **InitializeComponent**-Methode, und legen Sie die **ServiceName**-Eigenschaft auf **RemotingHost** fest.

```
this.ServiceName = "RemotingHost";
```

7. Suchen Sie die **OnStart**-Methode, und fügen Sie die folgende Codezeile zum Konfigurieren von Remoting hinzu. Der vollqualifizierte Pfad zur Konfigurationsdatei wird als Startparameter an den Dienst übergeben.

```
RemotingConfiguration.Configure(args[0]);
```

8. Fügen Sie dem Projekt eine neue C#-Klassendatei mit Namen **HostServiceInstaller** hinzu.
9. Fügen Sie einen Assemblyverweis auf die Assembly **System.Configuration.Install.dll** hinzu.
10. Fügen Sie am Anfang von **HostServiceInstaller** unterhalb der vorhandenen **using**-Anweisung die folgenden **using**-Anweisungen hinzu.

```
using System.ComponentModel;
using System.ServiceProcess;
using System.Configuration.Install;
```

11. Leiten Sie die **HostServiceInstaller**-Klasse von der **Installer**-Klasse ab.

```
public class HostServiceInstaller : Installer
```

12. Fügen Sie das **RunInstaller**-Attribut auf Klassenebene wie folgt hinzu:

```
[RunInstaller(true)]
public class HostServiceInstaller : Installer
```

13. Fügen Sie der **HostServiceInstaller**-Klasse die beiden folgenden Membervariablen vom Typ **private** hinzu. Diese Objekte werden beim Installieren des Dienstes verwendet.

```
private ServiceInstaller HostInstaller;
private ServiceProcessInstaller HostProcessInstaller;
```

14. Fügen Sie dem Konstruktor der **HostServiceInstaller**-Klasse den folgenden Code hinzu.

```
HostInstaller = new ServiceInstaller();
HostInstaller.StartType = System.ServiceProcess.ServiceStartMode.Manual;
HostInstaller.ServiceName = "RemotingHost";
HostInstaller.DisplayName = "Calculator Host Service";
Installers.Add (HostInstaller);
HostProcessInstaller = new ServiceProcessInstaller();
HostProcessInstaller.Account = ServiceAccount.User;
Installers.Add (HostProcessInstaller);
```

15. Klicken Sie im Projektmappen-Explorer mit der rechten Maustaste auf **RemotingHost**, zeigen Sie auf **Hinzufügen**, und klicken Sie dann auf **Neues Element hinzufügen**.
16. Klicken Sie in der Liste **Vorlagen** auf **Textdatei**, und nennen Sie die Datei **app.config**. Konfigurationsdateien mit dem Namen **app.config** werden von Visual Studio .NET als Teil des Erstellungsprozesses automatisch in den Ausgabeordner (z. B. <*Projektverzeichnis*>**\bin\debug**) kopiert und in <*Anwendungsname*>**.config** umbenannt.
17. Klicken Sie auf **OK**, um die neue Konfigurationsdatei hinzuzufügen.
18. Fügen Sie der neuen Konfigurationsdatei die folgenden Konfigurationselemente hinzu.

```
<configuration>
<system.runtime.remoting>
  <application name="RemoteHostService">
    <service>
      <wellknown type="RemoteObject.Calculator, RemoteObject"
                 objectUri="RemoteObject.Calculator" mode="Singleton" />
    </service>
    <channels>
      <channel ref="tcp" port="8085">
        <serverProviders>
          <formatter ref="binary" />
        </serverProviders>
      </channel>
    </channels>
  </application>
</system.runtime.remoting>
</configuration>
```

19. Klicken Sie im Menü **Erstellen** auf **Projektmappe erstellen**.

3. Erstellen eines Windows-Kontos zum Ausführen des Dienstes

Mit diesem Verfahren wird ein Windows-Konto erstellt, das zum Ausführen des Windows-Dienstes verwendet wird.

▶ **So erstellen Sie ein Windows-Kontos zum Ausführen des Dienstes**

1. Erstellen Sie ein neues lokales Benutzerkonto mit Namen **RemotingAccount**. Geben Sie ein Kennwort ein, und aktivieren Sie das Kontrollkästchen **Kennwort läuft nie ab**.
2. Klicken Sie in der Programmgruppe **Verwaltung** auf **Lokale Sicherheitsrichtlinie**.
3. Weisen Sie dem neuen Konto mithilfe des Tools **Lokale Sicherheitsrichtlinie** das Recht **Als Dienst anmelden** zu.

4. Installieren des Windows-Dienstes

In diesem Verfahren wird der Windows-Dienst unter Verwendung des Dienstprogramms **Installutil.exe** installiert und anschließend gestartet.

▶ **So installieren Sie den Windows-Dienst**

1. Öffnen Sie ein Befehlsfenster, und wechseln Sie in das Verzeichnis **Bin\Debug** im Projektordner **RemotingHost**.
2. Führen Sie das Dienstprogramm **Installutil.exe** aus, um den Dienst zu installieren.

   ```
   installutil.exe remotinghost.exe
   ```

3. Geben Sie im Dialogfeld **Dienstanmeldung festlegen** den Benutzernamen und das Kennwort des in Verfahren 3 erstellen Kontos ein, und klicken Sie auf **OK**.

   Zeigen Sie die Ausgabe des Dienstprogramms **Installutil.exe** an, und vergewissern Sie sich, dass der Dienst ordnungsgemäß installiert wurde.
4. Kopieren Sie die Assembly **RemoteObject.dll** in das Ausgabeverzeichnis des **RemotingHost**-Projekts (d. h. nach **RemotingHost\Bin\Debug**).
5. Starten Sie in der Programmgruppe **Verwaltung** das MMC-Snap-In Dienste.
6. Klicken Sie in der Liste **Dienste** mit der rechten Maustaste auf **Calculator Host Service**, und klicken Sie dann auf **Eigenschaften**.
7. Geben Sie den vollständigen Pfad zur Konfigurationsdatei des Dienstes (**remotinghost.exe.config**) in das Feld **Startparameter** ein.

   Hinweis: Dies ist im Handumdrehen erledigt, wenn Sie den Inhalt des Feldes **Pfad zur EXE-Datei** kopieren und diesen in das Feld **Startparameter** einfügen. Fügen Sie anschließend die Zeichenfolge ".config" an.
8. Klicken Sie auf **Starten**, um den Dienst zu starten.
9. Vergewissern Sie sich, dass sich der Status des Dienstes in **Gestartet** ändert.
10. Klicken Sie auf **OK**, um das Dialogfeld **Eigenschaften** zu schließen.

5. Erstellen einer Test-Clientanwendung

Mit diesem Verfahren wird eine Testkonsolenanwendung erstellt, die zum Aufrufen des Remoteobjekts innerhalb des Windows-Dienstes verwendet wird.

▶ **So erstellen Sie eine Test-Clientanwendung**

1. Fügen Sie dem aktuellen Projekt eine neue Visual C#-Konsolenanwendung mit Namen **RemotingClient** hinzu.
2. Klicken Sie im Projektmappen-Explorer mit der rechten Maustaste auf **RemotingClient**, und klicken Sie dann auf **Als Startprojekt festlegen**.

3. Fügen Sie einen Assemblyverweis auf die Assembly **System.Runtime.Remoting.dll** hinzu.
4. Fügen Sie dem **RemoteObject**-Projekt einen Projektverweis hinzu.
5. Fügen Sie am Anfang von **Class1.cs** unterhalb der vorhandenen **using**-Anweisungen die folgenden **using**-Anweisungen hinzu.

```
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
using RemoteObject;
```

6. Fügen Sie der **Main**-Methode den folgenden Testcode hinzu, um das **Calculator**-Objekt aufzurufen, für das der Windows-Dienst als Host fungiert.

```
TcpChannel chan = new TcpChannel();
ChannelServices.RegisterChannel(chan);
Calculator calc = (Calculator)Activator.GetObject(
                        typeof(RemoteObject.Calculator),
                       "tcp://localhost:8085/RemoteObject.Calculator");
if (calc == null)
  System.Console.WriteLine("Could not locate server");
else
  Console.WriteLine("21 + 21 is : " + calc.Add(21,21) );
```

7. Klicken Sie im Menü **Erstellen** auf **Projektmappe erstellen**.
8. Führen Sie die Clientanwendung aus, und vergewissern Sie sich, dass im Konsolenausgabefenster das korrekte Ergebnis angezeigt wird.

Referenzen

Informationen über das Hosten eines Remoteobjekts in ASP.NET (mit IIS) finden Sie in der Microsoft Knowledge Base im Artikel Q312107, "HOW TO: Host a Remote Object in IIS" (US).