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

Mit Enterprise Services (COM+) werden Rollen zum Zwecke der Autorisierung bereitgestellt. Nachstehend wird erläutert, wie eine Serviced Component für die rollenbasierte Sicherheit auf Methodenebene erstellt und konfiguriert wird.

Vorgehensweise: Verwenden von rollenbasierter Sicherheit mit Enterprise Services

In dieser Vorgehensweise wird erläutert, wie eine einfache Serviced Component erstellt wird, die ES-Rollen (Enterprise Services) für die Autorisierung verwendet.

Anmerkungen

- ES-Rollen sind nicht das Gleiche wie .NET-Rollen.
- ES-Rollen können Windows-Gruppen- oder –Benutzerkonten enthalten.
- ES-Rollen werden im COM+-Katalog verwaltet.
- ES-Rollen können auf der (ES-)Anwendungsebene, auf Schnittstellen-, Klassen- oder Methodenebene zugewiesen werden.
- ES-Rollen können teilweise deklarativ unter Verwendung von .NET-Attributen in der Assembly der Serviced Component konfiguriert werden.
- Windows-Gruppen- und Benutzerkonten müssen von einem Administrator zur Entwicklungszeit hinzugefügt werden.
- Administratoren können das Verwaltungsprogramm oder -skript für Komponentendienste verwenden.
- Für einen effektiven Einsatz der von Enterprise Services gebotenen rollenbasierten Sicherheit von einer ASP.NET-Webanwendung aus muss die Webanwendung die Windows-Authentifizierung verwenden und die Identität der Aufrufer vor dem Aufruf der Serviced Components wechseln.

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® Visual Studio® .NET als Entwicklungssystem

Die in dieser Vorgehensweise erläuterten Verfahren setzen zudem Kenntnisse der ASP.NET-Webentwicklung mit dem Entwicklungstool Microsoft Visual C#™ voraus.

Zusammenfassung

Diese Vorgehensweise enthält folgende Verfahren:

1. Erstellen einer C#-Klassenbibliotheksanwendung als Host für die Serviced Component
2. Erstellen der Serviced Component
3. Konfigurieren der Serviced Component
4. Erzeugen eines starken Namens für die Assembly
5. Erstellen der Assembly und Hinzufügen zum globalen Assemblycache
6. Manuelles Registrieren der Serviced Component
7. Prüfen der konfigurierten Anwendung
8. Erstellen einer Test-Clientanwendung

1. Erstellen einer C#-Klassenbibliotheksanwendung als Host für die Serviced Component

Mit dem nachstehenden Verfahren wird eine neue C#-Klassenbibliotheksanwendung erstellt, die die Serviced Component enthält.

▶ **So erstellen Sie eine C#-Klassenbibliotheksanwendung als Host für die Serviced Component**

1. Starten Sie Visual Studio .NET, und erstellen Sie in C# eine neue Klassenbibliotheksanwendung mit Namen **ServicedCom**.
2. Benennen Sie die standardmäßige Klassendatei **Class1.cs** in **SimpleComponent.cs** um.
3. Doppelklicken Sie auf **SimpleComponent.cs**, um die Datei zu öffnen, und benennen Sie den Typ **Class1** in **SimpleComponent** um. Aktualisieren Sie auch den Namen des Standardkonstruktors der Klasse entsprechend.

2. Erstellen der Serviced Component

In diesem Verfahren wird die **SimpleComponent**-Klasse von der **EnterpriseServices.ServicedComponent**-Klasse abgeleitet, um diesen Typ in eine Serviced Component umzuwandeln. Anschließend wird ein Schnittstelle erstellt und diese in der **SimpleComponent**-Klasse implementiert. Um Sicherheit auf Schnittstellen- und Methodenebene gewährleisten zu können, müssen Sie Schnittstellen definieren und implementieren.

▶ **So erstellen Sie die Serviced Component**

1. Fügen Sie einen Verweis auf die **System.EnterpriseServices**-Assembly hinzu.
2. Fügen Sie am Anfang der Datei **SimpleComponent.cs** unterhalb der vorhandenen **using**-Anweisungen die folgende **using**-Anweisung hinzu.

```
using System.EnterpriseServices;
```

3. Leiten Sie die **SimpleComponent**-Klasse von **ServicedComponent** ab.

```
public class SimpleComponent : ServicedComponent
```

4. Fügen Sie innerhalb des **ServicedCom**-Namespaces die folgende Schnittstellendefinition hinzu.

```
public interface ISomeInterface
{
  int Add( int operand1, int operand2 );
}
```

5. Leiten Sie **SimpleComponent** von dieser Schnittstelle ab.

```
public class SimpleComponent : ServicedComponent, ISomeInterface
```

6. Implementieren Sie die Schnittstelle wie folgt in der **SimpleComponent**-Klasse:

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

3. Konfigurieren der Serviced Component

Mit diesem Verfahren wird die Serviced Component für die rollenbasierte Sicherheit auf Methodenebene konfiguriert.

▶ **So konfigurieren Sie die Serviced Component**

1. Fügen Sie die folgenden Attribute direkt oberhalb der **SimpleComponent**-Klasse hinzu. Das **ComponentAccessControl**-Attribut ermöglicht Zugriffsüberprüfungen auf Komponentenebene, und das **SecureMethod**-Attribut ermöglicht Zugriffsüberprüfungen auf Methodenebene.

```
[ComponentAccessControl]
[SecureMethod]
public class SimpleComponent : ServicedComponent, ISomeInterface
```

2. Fügen Sie oberhalb der **Add**-Methode das folgende Attribut hinzu, um die **Manager**-Rolle zu erstellen und diese der Methode zuzuordnen.

```
[SecurityRole("Manager")]
public int Add( int operand1, int operand2 )
{
  return operand1 + operand2;
}
```

3. Öffnen Sie **Assemblyinfo.cs**, und fügen Sie am Anfang der Datei unterhalb der vorhandenen **using**-Anweisungen die folgende **using**-Anweisung hinzu.

```
using System.EnterpriseServices;
```

4. Wechseln Sie zum Ende der Datei, und fügen Sie die nachstehenden Attribute hinzu. Diese werden verwendet, um die Enterprise Services-Anwendung zu konfigurieren, die als Host für die Serviced Component dient.

```
// Configure the application as a server (out-of-process) application
[assembly: ApplicationActivation(ActivationOption.Server)]
// For meaningful role-based security, enable access checking at the process
// and component levels by using the following .NET attribute.
[assembly: ApplicationAccessControl(AccessChecksLevel=
                        AccessChecksLevelOption.ApplicationComponent)]
// Set the name and description for the application
[assembly: ApplicationName("SimpleRoles")]
[assembly: Description("Simple application to show ES Roles")]
// Add some additional roles
[assembly:SecurityRole("Employee")]
[assembly:SecurityRole("Senior Manager")]
```

4. Erzeugen eines starken Namens für die Assembly

Assemblys, die als Host für Serviced Components fungieren, müssen über einen starken Namen verfügen. Mit diesem Verfahren wird ein Schlüsselpaar aus öffentlichem und privatem Schlüssel erzeugt, das für die Vergabe eines starken Namens für die Assembly herangezogen wird.

▶ **So erzeugen Sie einen starken Namen für die Assembly**

1. Öffnen Sie ein Befehlsfenster, und wechseln Sie in das aktuelle Projektverzeichnis.
2. Erzeugen Sie mithilfe des Dienstprogramms **Sn.exe** eine Schlüsseldatei, die ein Schlüsselpaar bestehend aus einem öffentlichen und einem privaten Schlüssel enthält.

   ```
   sn.exe -k SimpleComponent.snk
   ```

3. Öffnen Sie **Assemblyinfo.cs** in Visual Studio.
4. Suchen Sie das Attribut [**AssemblyKeyFile**], und ändern Sie es so, dass es auf die Schlüsseldatei im Projektverzeichnis verweist:

   ```
   [assembly: AssemblyKeyFile(@"..\..\SimpleComponent.snk")]
   ```

5. Erstellen der Assembly und Hinzufügen zum globalen Assemblycache

In diesem Verfahren wird die Assembly erstellt, die die Serviced Component enthält. Anschließend wird die Assembly dem globalen Assemblycache hinzugefügt. Serviced Components sollten generell im globalen Assemblycache registriert werden, da es sich um Ressourcen auf Systemebene handelt. Im Gegensatz zu Bibliotheksanwendungen (für die diese Vorgehensweise allerdings ebenfalls zu empfehlen ist) MÜSSEN Serviced Components, für die COM+-Serveranwendungen als Host fungieren, im globalen Assemblycache installiert werden.

▶ **So erstellen Sie die Assembly und fügen sie dem globalen Assemblycache hinzu**

1. Klicken Sie im Menü **Erstellen** auf **Projektmappe erstellen**.
2. Kehren Sie zum Befehlsfenster zurück, und führen Sie den folgenden Befehl aus, um die Assembly dem globalen Assemblycache hinzuzufügen.

   ```
   gacutil –i bin\debug\ServicedCom.dll
   ```

6. Manuelles Registrieren der Serviced Component

Serviced Components können entweder manuell unter Verwendung des Tools **Regsvcs.exe** oder automatisch mithilfe der "verzögerten" Registrierung registriert werden. Bei der "verzögerten" Registrierung wird die Komponente (und die als Host fungierende COM+-Anwendung, die mithilfe der Metadaten der Assembly entwickelt und konfiguriert wurde) bei der erstmaligen Erstellung einer Instanz der Serviced Component registriert.

Um den – allerdings einmaligen – Leistungsabfall zu vermeiden, der mit diesem Ansatz einhergeht, wird die Serviced Component in diesem Verfahren manuell registriert.

So registrieren Sie die Serviced Component manuell

1. Kehren Sie zum Befehlsfenster zurück.
2. Führen Sie **Regsvcs.exe** aus, um die Komponente zu registrieren.

```
regsvcs bin\debug\ServicedCom.dll
```

7. Prüfen der konfigurierten Anwendung

In diesem Verfahren wird das Tool für Komponentendienste verwendet, um die Katalogeinstellungen zu prüfen, die als Ergebnis der zuvor verwendeten .NET-Attribute erstellt wurden.

▶ **So prüfen Sie die konfigurierte Anwendung**

1. Starten Sie die Komponentendienste aus der Programmgruppe **Verwaltung**.
2. Erweitern Sie die Ordner **Komponentendienste**, **Computer**, **Arbeitsplatz** und **COM+-Anwendungen**.
3. Klicken Sie mit der rechten Maustaste auf **SimpleRoles**, und klicken Sie dann auf **Eigenschaften**.
4. Klicken Sie auf die Registerkarte **Sicherheit**, und vergewissern Sie sich, dass **Zugriffsüberprüfung für diese Anwendung erzwingen** aktiviert ist und dass die Sicherheitsebene so eingestellt ist, dass Zugriffsüberprüfungen auf Prozess- und Komponentenebene durchgeführt werden. Diese Konfiguration ist das Ergebnis der im Vorfeld zugewiesenen .NET-Attribute.
5. Klicken Sie auf **OK**, um das Dialogfeld **Eigenschaften** zu schließen.
6. Erweitern Sie die Anwendung **SimpleRoles**, und erweitern Sie anschließend den Ordner **Components** und die **ServicedCom.SimpleComponent**-Klasse.
7. Suchen Sie die **Add**-Methode unterhalb der **ISomeInterface**-Methode im Ordner **Interfaces**.
8. Klicken Sie mit der rechten Maustaste auf **Add**, und klicken Sie dann auf **Eigenschaften**.
9. Klicken Sie auf die Registerkarte **Sicherheit**. Beachten Sie, dass die **Manager**-Rolle dieser Methode zugeordnet ist.
10. Klicken Sie auf **OK**, um das Dialogfeld **Eigenschaften** zu schließen.
11. Erweitern Sie den Ordner **Roles** unterhalb der **SimpleRoles**-Anwendung. Beachten Sie die Rollen, die Sie im Vorfeld mit .NET-Attributen erstellt haben. Beachten Sie zudem die **Marshaler**-Rolle. Diese wird als ein direktes Ergebnis des zuvor hinzugefügten [**SecureMethod**]-Attributs erstellt und ist für die Sicherheit auf Methodenebene zwingend erforderlich.

8. Erstellen einer Test-Clientanwendung

In diesem Verfahren wird eine Test-Clientanwendung basierend auf Windows-Formularen erstellt, um die Serviced Component zu instanziieren und aufzurufen.

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

1. Fügen Sie dem aktuellen Projekt mit C# eine neue Windows-basierte Anwendung mit Namen **TestClient** hinzu.
2. Fügen Sie dem **ServicedCom**-Projekt einen neuen Projektverweis hinzu.
   a. Klicken Sie im Projektmappen-Explorer mit der rechten Maustaste auf **Verweise**, und klicken Sie dann auf **Verweis hinzufügen**.
   b. Klicken Sie auf die Registerkarte **Projekte**.
   c. Wählen Sie **ServicedCom**, und klicken Sie auf **Auswählen** und dann auf **OK**.
3. Fügen Sie einen Verweis auf **System.EnterpriseServices** hinzu.
4. Fügen Sie dem Hauptformular der Anwendung eine Schaltfläche hinzu.
5. Doppelklicken Sie auf die Schaltfläche, um einen Ereignishandler für das Klickereignis der Schaltfläche zu erstellen.
6. Fügen Sie am Anfang von **Form1.cs** unterhalb der vorhandenen **using**-Anweisungen die folgende **using**-Anweisung hinzu.

```
using ServicedCom;
```

7. Kehren Sie zum Ereignishandler für das Klickereignis zurück, und fügen Sie den folgenden Code hinzu, um die Serviced Component zu instanziieren und aufzurufen.

```
SimpleComponent comp = new SimpleComponent();
MessageBox.Show( "Result is: " + comp.Add(1, 2));
```

8. Klicken Sie im Menü **Erstellen** auf **Projektmappe erstellen**.
9. Klicken Sie im Projektmappen-Explorer mit der rechten Maustaste auf das Projekt **TestClient**, und klicken Sie dann auf **Als Startprojekt festlegen**.
10. Drücken Sie **STRG+F5**, um die Anwendung **TestClient** auszuführen.
    Nun sollte eine unbehandelte Ausnahme erzeugt werden.
11. Klicken Sie im Meldungsfenster auf die Schaltfläche **Details**, um weitere Informationen zu der Ausnahmebedingung anzuzeigen.
    Wie Sie sehen, wurde eine **System.UnauthorizedAccessExecption** erzeugt. Der Grund hierfür ist, dass Ihr interaktives Anmeldekonto, das zum Ausführen der Anwendung **TestClient** verwendet wurde, kein Mitglied der **Manager**-Rolle ist, was jedoch unabdingbar ist, um die **Add**-Methode für die Serviced Component aufzurufen.
12. Klicken Sie auf **Quit**, um die Anwendung zu beenden.
13. Kehren Sie zu den Komponentendiensten zurück, und fügen Sie Ihr aktuelles (interaktives) Konto der **Manager**- und der **Marshaler**-Rolle hinzu.

Hinweis: Die Enterprise Services-Infrastruktur verwendet eine Reihe von Schnittstellen auf Systemebene, die von allen Serviced Components bereitgestellt werden. Hierzu gehören **IManagedObject**, **IDisposable** und **IServiceComponentInfo**. Wenn Zugriffsüberprüfungen auf Schnittstellen- oder Methodenebene aktiviert sind, wird der Enterprise Services-Infrastruktur der Zugriff auf diese Schnittstellen verweigert.

Infolgedessen erzeugt Enterprise Services eine spezielle Rolle mit Namen **Marshaler** und verbindet diese Rolle mit den Schnittstellen. Zur Bereitstellungszeit muss der Anwendungsadministrator also alle Benutzer der **Marshaler**-Rolle hinzufügen, die Zugriff auf beliebige Methoden oder Schnittstellen dieser Klasse benötigen. Hierfür gibt es zwei unterschiedliche Automatisierungsmöglichkeiten:

1. Schreiben Sie ein Skript, das das Objektmodell der Komponentendienste zugrunde legt, um alle Benutzer aus anderen Rollen in die **Marshaler**-Rolle zu kopieren.

2. Schreiben Sie ein Skript, dass alle anderen Rollen diesen drei speziellen Schnittstellen zuordnet, und löschen Sie die **Marshaler**-Rolle.

14. Schließen Sie die **SimpleRoles**-Anwendung, damit die Änderungen wirksam werden können. Klicken Sie hierzu auf den Namen der Anwendung, und klicken Sie dann auf **Shut down**.
15. Kehren Sie zu Visual Studio .NET zurück, und drücken Sie **STRG+F5**, um die **TestClient**-Anwendung erneut auszuführen.
16. Klicken Sie auf die entsprechende Schaltfläche im Formular, und vergewissern Sie sich, dass die Methode erfolgreich aufgerufen wird.