Identitätsmanagement mit ASP.NET

Inhaltsverzeichnis

Einleitung....1
Systemumgebung....2
Grundsätze von ASP.NET....2
Integrierte Authentifizierung....4
Providermodell....4
MembershipProvider....5
RoleProvider....5
Weitere verfügbare Implementierungen....6
Autorisierung....6
Dateien und Verzeichnisse....6
Anwendungscode....7
CardSpace und OpenID....8
Fazit....9
Quellen....10

Einleitung

Identitätsmanagement ist ein immer wichtiger werdendes Thema in der Webentwicklung. Dieses Thema wurde auch in Microsofts ASP.NET in der Version 2.0 umfangreich beachtet. War in ASP.NET und vorher in ASP noch kaum Unterstützung für das Identitätsmanagement vorhanden, so werden in ASP.NET 2.0 umfangreiche Funktionalitäten zur Verfügung gestellt.

Diese Ausarbeitung beschäftigt sich mit den wichtigsten Aspekten des Identitätsmanagement in ASP.NET in der Version 2.0. Sofern nicht anders benannt, bezieht sich im Folgenden „ASP.NET“ immer auf die Version 2.0.

Systemumgebung

ASP.NET wurde als Erweiterung zu den Microsoft Internet Information Services (IIS) entwickelt. Die übliche Systemumgebung besteht demzufolge mindestens aus einem Microsoft Windows Server Betriebssystem wie zum Beispiel Windows Server 2003, der dazugehörigen Version der Internet Information Services. Außerdem können weitere Komponenten, wie zum Beispiel ein Active Directory Service Teil der Umgebung sein.
Allerdings sind auch weniger typische Systemumgebungen möglich. So existiert als alternativer Webserver unter Windows der Cassini Webserver, der für die Entwicklung von ASP.NET-Anwendungen entwickelt wurde. Dieser besitzt aber keine Produktionsqualität und unterstützt auch nicht alle Funktionen wie die Internet Information Services, sodass dieser nicht in Produktionsumgebungen eingesetzt werden sollte. Der Cassini Webserver nutzt dieselbe ASP.NET-Implementierung wie die Internet Information Services, da beide auf das gleiche .NET-Framework setzen, dessen Bestandteil ASP.NET ist.
Außerdem steht noch das Open-Source-Projekt „Mono“ zur Verfügung, welches eine eigene .NET-Implementierung zur Verfügung stellt. Diese enthält eine relativ vollständige Implementierung von ASP.NET. Mono ist für viele Plattformen wie Unix, Linux, BSD, Solaris und Mac OS X verfügbar.

Grundsätze von ASP.NET

ASP.NET-Anwendungen bestehen hauptsächlich aus mehreren Dateien, auch alle Konfigurationen sind in Dateien gespeichert. Für jede ASP.NET-Anwendung existiert ein Wurzelverzeichnis. In diesem sind im Normalfall verschiedene .aspx-Dateien und eine Konfigurationsdatei mit dem Namen web.config vorhanden. Außerdem existieren verschiedene spezielle Unterordner.
Der wichtigste Dateityp sind .aspx-Dateien. Diese basieren auf XHTML-Dateien, sind aber um verschiedene Direktiven erweitert. Im Dateikopf einer solchen Datei steht in der page-Direktive die Programmiersprache, in der die Datei geschrieben wurde, und möglicherweise noch andere Optionen. Eine dieser Optionen ist CodeFile, mit der sich Code in eine andere Datei auslagern lässt, sodass dieser nicht im Kopf der Datei stehen muss (Code-Behind-Prinzip). Damit lässt sich die Darstellung etwas besser vom der Programmlogik trennen. Des Weiteren können in .aspx-Dateien ASP.NET-Controls eingebunden werden, die zur Laufzeit in

XHTML-Elemente umgewandelt werden. Wird beim Aufruf einer URL im Browser nur ein Verzeichnis angegeben, jedoch keine Datei, so wird in der Standardkonfiguration nach einer Datei namens „default.aspx“ gesucht und diese angezeigt. Hier einfaches Beispiel für eine .aspx-Datei:

```
<%@ Page Language="C#" CodeFile="Default.aspx.cs" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Einfache Labelseite</title>
</head>
<body>
    <form id="form1" runat="server">
        <asp:Label ID="lbl" runat="server" /> <br />
    </form>
</body>
</html>
```

Ein weiterer wichtiger Dateityp sind die .config-Dateien. Hierbei handelt es sich um spezielle XML-Dateien, in denen die Konfiguration für ASP.NET-Anwendungen gespeichert ist. Auf jedem ASP.NET-Server gibt es eine globale machine.config in der die rechnerspezifische Konfiguration, wie z.B. Datenbankverbindungen, gespeichert sind. Außerdem kann in jedem Anwendungsverzeichnis noch eine web.config liegen, die die anwendungsspezifische Konfiguration aufnimmt. Die Konfiguration wird vererbt, und müssen daher nicht unnötig oft geschrieben werden. Auch die Konfiguration für Authentifizierung und Autorisierung ist in den .config-Dateien hinterlegt. Ein Beispiel für eine web.config-Datei:

```
<?xml version="1.0"?>
<configuration>
   <appSettings/>
   <connectionStrings/>
   <system.web>
       <compilation debug="true"/>
       <authentication mode="Windows"/>
   </system.web>
</configuration>
```

Jede ASP.NET-Anwendung kann spezielle Unterordner enthalten. Hierzu gehören „App_Code“ für Anwendungscode (.cs- und .vb-Dateien) und „App_Data“ für Anwendungsdaten, welches das einzig von der Anwendung schreibbare Verzeichnis seien sollte. Außerdem können noch Ordner wie „App_Browsers“,

„App_LocalResources“, „App_GlobalResources“, „App_WebReferences“ und „App_Themes“ existieren.
Die Standardkonfiguration von ASP.NET-Anwendungen gilt als relativ sicher. Wenn beispielsweise eine unbehandelte Ausnahme in einer ASP.NET-Anwendung auftritt, so wird eine allgemeine Fehlermeldung angezeigt. Nur wenn die Anfrage, die den Fehler auslöste, vom lokalen System kam, werden detaillierte Fehlerinformationen und Teile des Quellcodes angezeigt.

Integrierte Authentifizierung

In ASP.NET sind vier verschiedene Optionen zur Authentifizierung verfügbar. Die einfachste und standardmäßig eingestellte Variante ist die Option „None“. Hierbei stellt ASP.NET keine Funktionalität zur Authentifizierung oder Autorisierung zur Verfügung. Der Entwickler kann eine komplett eigene Lösung entwickeln bzw. insofern schon vorhanden, nutzen. Bei der „Passport“-Authentifizierung werden die Nutzer über Microsofts Passport Service authentifiziert. Da sich Passport im Internet nicht durchsetzen konnte, ist diese Option für die meisten Entwickler nicht relevant. Die dritte Option ist die „Windows“-Authentifizierung. Hierbei wird die Authentifizierung durch den Webserver, meist also die IIS, vorgenommen. ASP.NET übernimmt die Authentifizierung direkt.
Die vierte und wahrscheinlich interessanteste Option stellt die „Forms“-Authentifizierung dar. Dabei erfolgt die Authentifizierung durch HTML-Formulare. Hierfür sind verschiedene vordefinierte Controls verfügbar. Es ist aber auch einfach möglich, eigene Formulare zu erstellen. Nahezu alle Komponenten sind hierbei austauschbar.

Providermodell

ASP.NET stellt ein Providermodell zu Anpassung der Authentifizierung zur Verfügung. Im Mittelpunkt stehen dabei die beiden abstrakten Klassen MembershipProvider und RoleProvider. Von diesen kann der Entwickler eigene Klassen ableiten und so ein eigenes Verhalten implementieren.

MembershipProvider

Wichtigste Providerklasse in ASP.NET ist der MembershipProvider. Er stellt eine komplette Nutzerverwaltung zur Verfügung. Dazu gehört das Anlegen, Ändern und Löschen von Nutzern. Dabei sind zahlreiche Sicherheitsparameter verfügbar. Hierzu zählen eine minimale Passwortlänge und ein gefordertes Passwortmuster, die Möglichkeit zum Fordern eine Passwort-Reset-Frage und Einstellungen zur maximalen Anzahl an falschen Passworteingabeversuchen sowie deren Rücksetzungszeit. Außerdem kann automatisch auf eindeutige E-Mail-Adressen geprüft werden, und die Möglichkeit, Passwörter im Klartext abzufragen, kann kontrolliert werden.
In ASP.NET sind bereits mehrere MembershipProvider integriert. Hierzu zählt der SqlMemberShipProvider, der die Benutzerdatenbank auf einem SQL-Server ablegt. Es existiert ein dazugehöriges Tool zum Anlegen der Tabellen und gespeicherten Prozeduren.
Des Weiteren steht der ActiveDirectoryMembershipProvider zur Verfügung. Dieser hält die Informationen in einem Active Directory, optional auch im Active Directory Application Mode (ADAM). Sollte der ActiveDirectory-MembershipProvider überhaupt im Internet zum Einsatz kommen, so wird dies meist im Application Mode geschehen, der unabhängig von der Windows-Domäne des Servers arbeitet.

RoleProvider

Der zweite wichtige Provider ist der RoleProvider, der eine Rollenzuordnung für Benutzer zur Verfügung stellt. Mit ihm lassen sich Benutzer zu Rollen zuordnen und von diesen Entfernen. Außerdem kann beispielsweise geprüft werden, ob ein Benutzer in einer bestimmten Rolle ist, und welche Benutzer in einer Rolle sind.
Wie schon beim MembershipProvider stehen auch für den RoleProvider einige Implementierungen zur Verfügung. Analog zum SqlMembershipProvider steht ein SqlRoleProvider zur Verfügung, der die Rollenzuordnung auf einem SQL-Server hinterlegt. Des Weiteren ist der AuthorizationStoreRoleProvider verfügbar. Dieser kann die Rollen und deren Zuordnung zu Benutzern wahlweise in einer lokalen XML-Datei oder im Active Directory speichern, optional auch hier wieder im Active Directory Application Mode. Als dritte Implementierung existiert der

WindowsTokenRoleProvider, bei dem die Rollen den Gruppen der integrierten Windows-Sicherheit entsprechen.

Weitere verfügbare Implementierungen

Für das Providermodell stehen noch weitere Implementierungen zur Verfügung. Diese werden zum Teil von kommerziellen Anbietern angeboten und sind teilweise kostenlos und auch quelloffen verfügbar. Ein wichtiges Beispiel für eine Implementierung eines kommerziellen Anbieters sind die Oracle Providers for ASP.NET. Diese sind Teil der Oracle Data Access Components (ODAC) und bieten sowohl einen Membership- als auch einen RoleProvider.
Auf der Webseite des Code Project und in Microsofts Codeplex lassen sich weitere Implementierungen finden. So steht etwa die mySQL-Suite für das Speichern der Daten in mySQL-Datenbanken zur Verfügung. Außerdem ist mit den SQLiteProviders ein Speichern in lokalen SQLite-Datenbank-Dateien möglich. Auch für weitere Datenbanken gibt es entsprechende Provider.
Außerhalb der gewöhnliche RDBMS gibt es noch weitere Möglichkeiten zum halten der Benutzer- und Rollendaten. Die XmlProviderCollection ermöglicht das Lesen und Schreiben der Daten in XML-Dateien, die NHibernateProvider-Implementierungen sichern die Daten mithilfe des NHibernate-Persistenz-Frameworks. Eine weitere Implementierung stellen die WSSecurityProvider dar. Hierdurch lassen sich die Daten via SOAP/XML übertragen. Dabei sollte natürlich auf eine sichere Kommunikation mit dem Webservice geachtet werden.

Autorisierung

Dateien und Verzeichnisse

In ASP.NET gibt es verschiedene Wege, um eine Autorisierung durchzuführen. Die einfachste, aber eine dennoch effektive Variante ist das Absichern einzelner Dateien oder Verzeichnisse bzw. deren URLs mithilfe der ASP.NET-Konfigurationsdateien. Dazu werden im authorization-Teil eines location-Elements der web.config die entsprechenden Rechte vergeben. Dabei wird der Zugriff auf eine Ressource entweder verhindert (deny) oder erlaubt (allow). Dabei kann man für jede beliebige Kombination aus Benutzername, Rolle und HTTP-Verb einen entsprechenden Eintrag hinzufügen. Als Vereinfachung stehen die beiden besonderen Benutzernamen * und ? zur Verfügung, die alle Benutzer (*) oder

nicht authentifizierte Benutzer (?) beschreiben. Hierdurch lassen sich relativ einfach abgesicherte Bereich zum Beispiel nur für authentifizierte Mitglieder definieren. Die Prüfung der Autorisierung findet in der Reihenfolge statt, die durch die Einträge in der Datei gegeben ist, wobei die Einträge auch vererbt werden. Ein einfaches Beispiel hierfür ist die folgende Konfiguration:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <location path="/information">
        <system.web>
            <authorization>
                <allow VERB="POST" users="John, Turk" />
                <deny VERB="POST" users="*" />
                <allow VERB="GET" users="*" />
            </authorization>
        </system.web>
    </location>
</configuration>
```

In diesem Beispiel dürfen nur die Benutzer „John“ und „Turk“ die HTTP-POST-Methode auf Ressourcen im Pfad „/information“ anwenden, während alle Benutzer die Methode GET benutzen dürfen.
Bei der Absicherung von Dateien und Verzeichnissen sollte beachtet werden, dass in der Standardkonfiguration nur ASP.NET-Dateien gesichert werden. Die Ursache hierfür ist die Tatsache, dass ASP.NET ein ISAPI-Filter für die Internet Information Services ist, der im Normalfall nur ASP.NET-spezifische Dateien filtert. Daher sind weitere Ressourcen, wie zum Beispiel Bilder oder Textdateien nicht geschützt. Diese Einstellung lässt sich aber einfach durch eine entsprechende Konfiguration des ISAPI-Filters in der Verwaltung der Internet Information Services ändern.

Anwendungscode

ASP.NET biete weitere Möglichkeit zur Autorisierung. Mithilfe dieser Methode kann man den Zugriff von Benutzern und Rollen auf einzelne Klassen oder Funktionen für kontrollieren. Ermöglicht wird das durch die Attribute im .NET-Framework. So ist es möglich, eine Klasse oder Funktion mit dem Attribut PrincipalPermission zu markieren. Damit lässt sich dann zum Beispiel definieren, dass ein bestimmter Benutzer die Funktion ausführen darf. Hierzu ein einfaches Beispiel:

```
[PrincipalPermission(SecurityAction.Demand, Role:="ImportantUsers")]
void DoSomethingImportant() {
    // ...
    return;
}

[PrincipalPermission(SecurityAction.Demand, Authenticated:=true)]
class SensitiveInformationHandler {
    // ...
}
```

In diesem Beispiel wird der Zugriff auf die Funktion DoSomethingImportant nur Benutzern gestattet, die in der Rolle „ImportantUsers“ sind. Der Zugriff auf die Klasse SensitiveInformationHandler ist nur authentifizierten Benutzern gestattet. Der Vorteil dieser Methode ist, dass gezielt die kritischen Methoden einer Webanwendung abgesichert werden können. So können beispielsweise Funktionen, die kritische Daten, wie zum Beispiel Datenbankverbindungen, anzeigen und ändern, nur für die Administratorrolle freigegeben werden. Ein Nachteil dieser Methode ist allerdings, dass die Autorisierung nicht zentral an einer Stelle vorgenommen wird, was die Verifizierung der Sicherheit und die dazugehörige Wartung erschwert.
Im Vergleich lässt sich mithilfe der URL-basierten Autorisierung eine übersichtliche und grundlegende Sicherheit schaffen, während sich mit der codebasierten Autorisierung die Sicherheit genauer kontrollieren lässt. Eine Kombination beider Methoden lässt eine grobgranulare Sicherheit mithilfe der URL-basierten Autorisierung zu, während einzelne besonders wichtige Funktionen oder Klassen nochmal gezielt gesichert werden können.

CardSpace und OpenID

Ein aktuelles Thema zum Identitätsmanagement sind CardSpace und OpenID als neue Technologien für die Authentifizierung. Beide Technologien bieten eine Authentifizierung abseits der üblichen Benutzername-Passwort-Variante. Für beide Methoden existiert bis jetzt allerdings kein MembershipProvider oder RoleProvider was in der Technik beider Technologien begründet ist. Sowohl CardSpace als auch OpenID ersetzen die Benutzername-Passwort-Kombination, eine Benutzerverwaltung auf Anwendungsseite ist aber meist trotzdem nötig. So existiert für OpenID ein frei verfügbares OpenID-Login-Control, was anstelle des

normalen ASP.NET-Login-Control eingesetzt werden kann. Dieses lässt sich dann weiterhin mit allen MembershipProvider-Implementierungen betreiben.
Für CardSpace bietet Microsoft selbst das „Information Card Kit for ASP.NET 2.0“ an. Damit lässt sich CardSpace in eigene ASP.NET Anwendungen, die dann als Relying Party agieren, integrieren.

Fazit

ASP.NET ist eine Webanwendungsumgebung, die für den Produktiveinsatz auch kritischer Anwendungen geeignet ist. ASP.NET bietet viele integrierte Basisfunktionen für Authentifizierung und Autorisierung, die sich aber auch leicht erweitern und ersetzen lassen. Gerade das Providermodell bietet einen interessanten Ansatz, der gut auf Anforderungen wie Wiederverwendbarkeit und Konfigurierbarkeit eingeht.
Ein Nachteil von ASP.NET ist die Abhängigkeit von Microsoft-Software. Zwar gibt es Implementierungen für andere Systeme, diese werden aber kaum genutzt und sind demnach auch nicht so gut getestet wie die Referenzimplementierung. Viele Systemadministratoren scheuen den Einsatz von Microsoft-Betriebssystem an sicherheitskritischen Stellen.

Quellen

- Wikipedia: ASP.NET (Version vom 19.1.2008)
  http://en.wikipedia.org/w/index.php?title=ASP.NET&oldid=185422518
- MSDN Library: ASP.NET Web Applications
  http://msdn2.microsoft.com/de-de/asp.net/ms644563.aspx
- Microsoft: Building Secure ASP.NET Applications
  http://msdn2.microsoft.com/en-us/library/aa302415.aspx
- Microsoft: Authentication in ASP.NET: .NET Security Guidance
  http://msdn2.microsoft.com/en-us/library/ms978378.aspx
- GotDotNet: Autorisieren von Benutzern und Rollen
  http://de.gotdotnet.com/quickstart/aspplus/doc/authorization.aspx
- GotDotNet: Formularbasierte Authentifizierung
  http://de.gotdotnet.com/quickstart/aspplus/doc/formsauth.aspx
- DotnetCoders: User Authorization in Sub-Directories
  http://www.dotnetcoders.com/web/Articles/ShowArticle.aspx?article=186
- OdeToCode: Membership and Role Providers in ASP.NET 2.0
  http://www.odetocode.com/Articles/427.aspx
- Alternative Provider:
  http://www.codeplex.com/XmlProviderLibrary
  http://www.codeplex.com/nhibernateprovider
  http://www.codeproject.com/aspnet/WSSecurityProvider.asp
  http://www.codeproject.com/aspnet/MySQLsuite.asp
  http://www.codeproject.com/useritems/SQLiteProviders.asp