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

Nachstehend wird erläutert, wie Sie die Formularauthentifizierung für einen Active Directory-Speicher für Anmeldeinformationen implementieren.

Vorgehensweise: Verwenden der Formular-authentifizierung mit Active Directory

Mit der von ASP.NET gebotenen Formularauthentifizierung können Benutzer sich identifizieren, indem sie Anmeldeinformationen (einen Benutzernamen und ein Kennwort) in ein Webformular eingeben. Beim Empfang dieser Anmeldeinformationen kann die Webanwendung den Benutzer authentifizieren, indem die Kombination aus Benutzername und Kennwort anhand einer Datenquelle geprüft wird.

Nachstehend wird erläutert, wie Benutzer anhand des Verzeichnisdienstes Microsoft® Active Directory® unter Verwendung von LDAP (Lightweight Directory Access Protocol) authentifiziert werden können. Darüber hinaus wird beschrieben, wie eine Liste der Sicherheitsgruppen sowie die Verteilerlisten abgerufen werden können, denen der Benutzer angehört, und wie diese Informationen in einem **GenericPrincipal**-Objekt gespeichert werden, das dann in der **HttpContext.Current.User**-Eigenschaft gespeichert wird, die mit der Anforderung die ASP.NET-Webanwendung durchläuft. Die Eigenschaft kann anschließend für die rollenbasierte Authentifizierung in .NET verwendet werden.

Anforderungen

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

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

Die in dieser Vorgehensweise erläuterten Verfahren setzen zudem Kenntnisse des Entwicklungstools Microsoft Visual C#™ voraus.

Zusammenfassung

Diese Vorgehensweise enthält folgende Verfahren:

1. Erstellen einer Webanwendung mit einer Anmeldeseite
2. Konfigurieren der Webanwendung für die Formularauthentifizierung
3. Entwickeln des LDAP-Authentifizierungscodes zum Nachschlagen des Benutzers in Active Directory
4. Entwickeln des LDAP-Gruppenabrufcodes zur Feststellung der Gruppenmitgliedschaft des Benutzers
5. Authentifizieren des Benutzers und Erstellen eines Formularauthentifizierungstickets
6. Implementieren eines Authentifizierungsanforderungshandlers zum Erstellen des **GenericPrincipal**-Objekts
7. Testen der Anwendung

1. Erstellen einer Webanwendung mit einer Anmeldeseite

Im Verlauf dieses Verfahrens wird eine einfache C#-Webanwendung programmiert, die eine Anmeldeseite enthält, auf der der Benutzer einen Benutzernamen und ein Kennwort eingeben kann. Des Weiteren ist eine Standardseite enthalten, auf der Informationen zur Identität (Name) und zur Gruppenmitgliedschaft im Zusammenhang mit der aktuellen Webanforderung angezeigt werden.

υ So erstellen Sie eine Webanwendung mit einer Anmeldeseite

1. Starten Sie Visual Studio .NET, und erstellen Sie eine neue C# ASP.NET-Webanwendung mit Namen **FormsAuthAD**.
2. Benennen Sie **WebForm1.aspx** mithilfe des Projektmappen-Explorers in **Logon.aspx** um.
3. Fügen Sie der **System.DirectoryServices.dll** einen neuen Assemblyverweis hinzu. Hiermit wird der Zugriff auf den Namespace **System.DirectoryServices** ermöglicht, der verwaltete Typen enthält, die das Abfragen und Bearbeiten in Active Directory vereinfachen.
4. Fügen Sie der Seite **Logon.aspx** die in Tabelle 1 aufgeführten Steuerelemente hinzu, um ein einfaches Anmeldeformular zu erstellen.

Tabelle 1: Die Steuerelemente für **Logon.aspx**

Typ des Steuerelements	Text	ID
Bezeichnungsfeld	Domain Name:	-
Bezeichnungsfeld	User Name:	-
Bezeichnungsfeld	Password	-
Textfeld	-	**txtDomainName**
Textfeld	-	**txtUserName**
Textfeld	-	**txtPassword**
Schaltfläche	Log On	**btnLogon**
Bezeichnungsfeld		**lblError**

5. Legen Sie die **TextMode**-Eigenschaft von **txtPassword** auf **Password** fest.
6. Klicken Sie im Projektmappen-Explorer mit der rechten Maustaste auf **FormsAuthAD**, zeigen Sie auf **Hinzufügen**, und klicken Sie dann auf **Web Form hinzufügen**.
7. Geben Sie im Feld **Name** den Namen **default.aspx** ein, und klicken Sie dann auf **Öffnen**.
8. Klicken Sie im Projektmappen-Explorer mit der rechten Maustaste auf **default.aspx**, und klicken Sie dann auf **Als Startseite festlegen**.

9. Doppelklicken Sie auf **default.aspx**, um den Ereignishandler zum Laden der Seite anzuzeigen.
10. Fügen Sie dem Ereignishandler den folgenden Code hinzu, um den mit der aktuellen Webanforderung verbundenen Namen der Identität anzuzeigen.

```
Response.Write( HttpContext.Current.User.Identity.Name );
```

2. Konfigurieren der Webanwendung für die Formular-authentifizierung

Mit dem folgenden Verfahren wird die Datei **Web.config** der Anwendung bearbeitet, um die Anwendung für die Formularauthentifizierung zu konfigurieren.

υ So konfigurieren Sie die Webanwendung für die Formularauthentifizierung

1. Öffnen Sie **Web.config** im Projektmappen-Explorer.
2. Suchen Sie das <**authentication**>-Element, und ändern Sie das **mode**-Attribut in **Forms**.
3. Fügen Sie das <**forms**>-Element als untergeordnetes Element des <**authentication**>-Elements hinzu, und legen Sie die Attribute **loginUrl**, **name**, **timeout** und **path** wie nachstehend gezeigt fest.

```
<authentication mode="Forms">
  <forms loginUrl="logon.aspx" name="adAuthCookie" timeout="60" path="/">
  </forms>
</authentication>
```

4. Fügen Sie das folgende <**authorization**>-Element unterhalb des <**authentication**>-Elements hinzu. Dies sorgt dafür, dass nur authentifizierte Benutzer auf die Anwendung zugreifen können. Das im Vorfeld eingerichtete **loginUrl**-Attribut des <**authentication**>-Elements leitet nicht authentifizierte Anforderungen auf die Seite **Logon.aspx** um.

```
<authorization>
  <deny users="?" />
  <allow users="*" />
</authorization>
```

5. Speichern Sie die Datei **Web.config**.
6. Starten Sie das IIS-MMC-Snap-In (Microsoft Management Console).
7. Klicken Sie mit der rechten Maustaste auf das virtuelle Verzeichnis der Anwendung und dann auf **Eigenschaften**.
8. Klicken Sie auf die Registerkarte **Verzeichnissicherheit**, und klicken Sie dann in der Gruppe **Authentifizierung und Zugriffssteuerung** auf die Schaltfläche **Bearbeiten**.
9. Aktivieren Sie das Kontrollkästchen **Anonymer Zugriff**, und deaktivieren Sie **Kennwortsteuerung durch IIS zulassen**.
10. Da das standardmäßige anonyme Konto IUSR_MACHINE nicht über die Berechtigung zum Zugriff auf Active Directory verfügt, erstellen Sie ein neues Konto mit minimalen Rechten und geben die Kontodetails im Dialogfeld **Authentifizierungsmethoden** ein.
11. Klicken Sie auf **OK** und dann erneut auf **OK**, um das Dialogfeld **Eigenschaften** zu schließen.
12. Kehren Sie zu Visual Studio .NET zurück, und fügen Sie in **Web.config** unterhalb des <**authorization**>-Elements ein <**identity**>-Element hinzu. Setzen Sie das Attribut für Identitätswechsel auf **true**. Dies bewirkt, dass ASP.NET die Identität des anonymen Kontos wie weiter oben erläutert wechselt.

```
<identity impersonate="true" />
```

Als Ergebnis dieser Konfiguration werden alle Anforderungen an die Anwendung unter dem Sicherheitskontext des konfigurierten anonymen Kontos ausgeführt. Der Benutzer stellt die Anmeldeinformationen für die Authentifizierung bei Active Directory über das Webformular bereit, für den Zugriff auf Active Directory wird jedoch das konfigurierte anonyme Konto verwendet.

3. Entwickeln des LDAP-Authentifizierungscodes zum Nachschlagen des Benutzers in Active Directory

Mit diesem Verfahren wird der Webanwendung eine neue Hilfsklasse hinzugefügt, um den LDAP-Code zu kapseln. Die Klasse stellt zunächst eine **IsAuthenticated**-Methode bereit, um eine übergebene Domäne, einen Benutzernamen und ein Kennwort anhand eines Active Directory-Benutzerobjekts zu prüfen.

υ So entwickeln Sie den LDAP-Authentifizierungscode zum Nachschlagen des Benutzers in Active Directory

1. Fügen Sie eine neue C#-Klassendatei mit Namen **LdapAuthentication.cs** hinzu.
2. Fügen Sie einen Verweis auf die Assembly **System.DirectoryServices.dll** hinzu.
3. Fügen Sie am Anfang von **LdapAuthentication.cs** die folgenden **using**-Anweisungen hinzu.

```
using System.Text;
using System.Collections;
using System.DirectoryServices;
```

4. Benennen Sie den vorhandenen Namespace in **FormsAuthAD** um.
5. Fügen Sie der **LdapAuthentication**-Klasse zwei Zeichenfolgen vom Typ **private** hinzu, eine für den LDAP-Pfad zu Active Directory und die andere für das Filterattribut, das zum Durchsuchen von Active Directory verwendet wird.

```
private string _path;
private string _filterAttribute;
```

6. Fügen Sie einen Konstruktor vom Typ **public** hinzu, der zum Initialisieren des Active Directory-Pfades verwendet werden kann.

```
public LdapAuthentication(string path)
{
  _path = path;
}
```

7. Fügen Sie die folgende **IsAuthenticated**-Methode hinzu, die einen Domänennamen, einen Benutzernamen und ein Kennwort als Parameter akzeptiert und **bool** zurückgibt, um anzugeben, ob ein Benutzer mit einem übereinstimmenden Kennwort in Active Directory vorhanden ist. Diese Methode versucht zunächst, die Bindung zu Active Directory unter Verwendung der übergebenen Anmeldeinformationen herzustellen. Im Erfolgsfall verwendet die Methode die verwaltete Klasse **DirectorySearcher**, um nach dem angegebenen Benutzerobjekt zu suchen. Falls der Benutzer gefunden werden kann, wird das **_path**-Member so aktualisiert, dass es auf das Benutzerobjekt verweist, und das **_filterAttribute**-Member wird mit dem CN-Attribut (Common Name, Gemeinsamer Name) des Benutzerobjekts aktualisiert.

```
public bool IsAuthenticated(string domain, string username, string pwd)
{
  string domainAndUsername = domain + @"\" + username;
  DirectoryEntry entry = new DirectoryEntry( _path,
                                             domainAndUsername, pwd);

  try
  {
    // Ruft das systemeigene ADSI-Objekt zur Authentifizierung ab.
    Object obj = entry.NativeObject;
    DirectorySearcher search = new DirectorySearcher(entry);
    search.Filter = "(SAMAccountName=" + username + ")";
    search.PropertiesToLoad.Add("cn");
    SearchResult result = search.FindOne();
    if(null == result)
    {
      return false;
    }
    // Aktualisierung der neuen Position des Benutzers im Verzeichnis
    _path = result.Path;
    _filterAttribute = (String)result.Properties["cn"][0];
  }
  catch (Exception ex)
  {
    throw new Exception("Error authenticating user. " + ex.Message);
  }
  return true;
}
```

4. Entwickeln des LDAP-Gruppenabrufcodes zur Feststellung der Gruppenmitgliedschaft des Benutzers

In diesem Verfahren wird die **LdapAuthentication**-Klasse erweitert, um eine **GetGroups**-Methode zum Abrufen einer Liste der Gruppen bereitzustellen, denen der aktuelle Benutzer angehört. Die **GetGroups**-Methode gibt die Gruppenliste als eine mit einem senkrechten Balken (Pipe, " | ") getrennte Zeichenfolge aus.

```
"Group1|Group2|Group3|"
```

υ So entwickeln Sie den LDAP-Gruppenabrufcode zum Feststellen der Gruppenmitgliedschaft des Benutzers

1. Fügen Sie der **LdapAuthentication**-Klasse die folgende Implementierung der **GetGroups**-Methode hinzu:

```
public string GetGroups()
{
  DirectorySearcher search = new DirectorySearcher(_path);
  search.Filter = "(cn=" + _filterAttribute + ")";
  search.PropertiesToLoad.Add("memberOf");
  StringBuilder groupNames = new StringBuilder();
  try
  {
    SearchResult result = search.FindOne();
    int propertyCount = result.Properties["memberOf"].Count;
    String dn;
    int equalsIndex, commaIndex;

    for( int propertyCounter = 0; propertyCounter < propertyCount;
         propertyCounter++)
    {
      dn = (String)result.Properties["memberOf"][propertyCounter];

      equalsIndex = dn.IndexOf("=", 1);
      commaIndex = dn.IndexOf(",", 1);
      if (-1 == equalsIndex)
      {
        return null;
      }
      groupNames.Append(dn.Substring((equalsIndex + 1),
                        (commaIndex - equalsIndex) - 1));
      groupNames.Append("|");
    }
  }
  catch(Exception ex)
  {
    throw new Exception("Error obtaining group names. " + ex.Message);
  }
  return groupNames.ToString();
}
```

5. Authentifizieren des Benutzers und Erstellen eines Formularauthentifizierungstickets

In diesem Verfahren wird der **btnLogon_Click**-Ereignishandler zum Authentifizieren von Benutzern implementiert. Für authentifizierte Benutzer erstellen Sie dann ein Formularauthentifizierungsticket, das die Gruppenliste des Benutzers enthält. Anschließend leiten Sie die Benutzer auf die Seite um, die ursprünglich (vor der Umleitung auf die Anmeldeseite) angefordert wurde.

- **So authentifizieren Sie den Benutzer und erstellen ein Formular-authentifizierungsticket**

1. Kehren Sie zum Formular **Logon.aspx** zurück, und doppelklicken Sie auf die Schaltfläche **Log On**, um einen leeren **btnLogon_Click**-Ereignishandler zu erstellen.
2. Fügen Sie am Anfang der Datei unterhalb der vorhandenen **using**-Anweisungen die folgende **using**-Anweisung hinzu. Hiermit wird der Zugriff auf die **FormsAuthentication**-Methoden ermöglicht.

```
using System.Web.Security;
```

3. Fügen Sie Code hinzu, um eine neue Instanz der **LdapAuthentication**-Klasse zu erstellen, die so initialisiert wird, dass sie auf das Active Directory von LDAP verweist, wie im folgenden Code gezeigt. Denken Sie daran, den Pfad so zu ändern, dass er auf den Active Directory-Server verweist.

```
// Pad zum LDAP-Verzeichnisdienst.
// Kontaktieren Sie Ihren Netzwerkadministrator, um einen gültigen Pfad
// zu erhalten.
string adPath = "LDAP://yourCompanyName.com/DC=yourCompanyName,DC=com";
LdapAuthentication adAuth = new LdapAuthentication(adPath);
```

4. Fügen Sie den folgenden Code hinzu, damit die nachstehenden Schritte ausgeführt werden:
   a. Authentifizieren des Aufrufers anhand von Active Directory.
   b. Abrufen der Liste der Gruppen, denen der Benutzer angehört.
   c. Erstellen eines **FormsAuthenticationTicket**, das die Gruppenliste enthält.
   d. Verschlüsseln des Tickets.
   e. Erstellen eines neuen Cookies, das das verschlüsselte Ticket enthält.
   f. Hinzufügen des Cookies zur Cookieauflistung, die an den Browser des Benutzers zurückgegeben wird.

```
try
{
  if(true == adAuth.IsAuthenticated(txtDomainName.Text,
                                    txtUserName.Text,
                                    txtPassword.Text))
  {
    // Ermittlung der Gruppen des Benutzers
    string groups = adAuth.GetGroups();
    // Erstellen eines Authentifizierungstickets
    FormsAuthenticationTicket authTicket =
        new FormsAuthenticationTicket(1,  // Version
                                      txtUserName.Text,
                                      DateTime.Now,
                                      DateTime.Now.AddMinutes(60),
                                      false, groups);
    // Verschlüsselung des Tickets.
    string encryptedTicket = FormsAuthentication.Encrypt(authTicket);
    // Erstellung eines Cookies und Hinzufügen der Ticketdaten zu diesem.
    HttpCookie authCookie =
                new HttpCookie(FormsAuthentication.FormsCookieName,
                               encryptedTicket);
    // Fügt der Cookieauflistung das angegebene Cookie hinzu.
    Response.Cookies.Add(authCookie);

    // Leitet den Benutzer zu der ursprünglich angeforderten Seite um.
    Response.Redirect(
              FormsAuthentication.GetRedirectUrl(txtUserName.Text,
                                                 false));
  }
```

```
  else
  {
    lblError.Text =
         "Authentication failed, check username and password.";
  }
}
catch(Exception ex)
{
  lblError.Text = "Error authenticating. " + ex.Message;
}
```

6. Implementieren eines Authentifizierungsanforderungs-handlers zum Erstellen des GenericPrincipal-Objekts

Mit diesem Verfahren wird der Ereignishandler **Application_AuthenticateRequest** in **global.asax** implementiert und ein **GenericPrincipal**-Objekt für den aktuell authentifizierten Benutzer erstellt. Dieses Objekt enthält die Liste der Gruppen, deren Mitglied der Benutzer ist, und die aus dem im Authentifizierungscookie enthaltenen **FormsAutheticationTicket** abgerufen wird. Schließlich ordnen Sie das **GenericPrincipal**-Objekt dem aktuellen **HttpContext**-Objekt zu, das für jede Webanforderung erstellt wird.

υ So implementieren Sie einen Authentifizierungsanforderungshandler zum Erstellen eines GenericPrincipal-Objekts

1. Öffnen Sie **gobal.asax.cs** im Projektmappen-Explorer.
2. Fügen Sie am Anfang der Datei die folgenden **using**-Anweisungen hinzu.

```
using System.Web.Security;
using System.Security.Principal;
```

3. Suchen Sie den Ereignishandler **Application_AuthenticateRequest**, und fügen Sie den folgenden Code hinzu, um aus der mit der Anforderung übergebenen Cookieauflistung das Cookie abzurufen, das das verschlüsselte **FormsAuthenticationTicket** enthält.

```
// Gibt den Cookienamen der aktuellen Anwendung zurück.
string cookieName = FormsAuthentication.FormsCookieName;
HttpCookie authCookie = Context.Request.Cookies[cookieName];

if(null == authCookie)
{
  // Kein Authentifizierungscookie gefunden.
  return;
}
```

4. Fügen Sie den folgenden Code hinzu, um das **FormsAuthenticationTicket** aus dem Cookie zu extrahieren und zu entschlüsseln.

```
FormsAuthenticationTicket authTicket = null;
try
{
  authTicket = FormsAuthentication.Decrypt(authCookie.Value);
}
catch(Exception ex)
{
  // Festhalten der Ausnahmedetails
  return;
}

if (null == authTicket)
{
  // Cookie konte nicht entschlüsselt werden.
  return;
}
```

5. Fügen Sie den folgenden Code hinzu, um die per Pipe getrennte Liste der Gruppennamen auszulesen, die bei der ursprünglichen Authentifizierung des Benutzers an das Ticket angehängt wurde.

```
// Bei der Ticketerstellung wurde der UserData-Eigenschaft eine durch Pipe-Zeichen
// getrennte Liste von Gruppennamen zugewiesen.
String[] groups = authTicket.UserData.Split(new char[]{'|'});
```

6. Fügen Sie den folgenden Code hinzu, um das **GenericIdentity**-Objekt mit dem Benutzernamen zu erstellen, der dem Ticketnamen entnommen wurde, und um ein **GenericPrincipal**-Objekt zu erstellen, das diese Identität zusammen mit der Gruppenliste des Benutzers enthält.

```
// Erstellen eines Identitätsobjektes
GenericIdentity id = new GenericIdentity(authTicket.Name,
                                         "LdapAuthentication");

// Der Principal wird an die Anforderung durchgereicht.
GenericPrincipal principal = new GenericPrincipal(id, groups);
// Anbindung des neuen Principals an das aktuelle HttpContext-Objekt.
Context.User = principal;
```

7. Testen der Anwendung

In diesem Verfahren wird die Seite **default.aspx** mithilfe der Webanwendung angefordert. Sie werden zwecks Authentifizierung auf die Anmeldeseite umgeleitet. Nach erfolgreicher Authentifizierung wird der Browser auf die ursprünglich angeforderte Seite **default.aspx** umgeleitet. Hierbei wird die Liste der Gruppen, denen der authentifizierte Benutzer angehört, aus dem **GenericPrincipal**-Objekt extrahiert und angezeigt. Dieses **GenericPrincipal**-Objekt wurde vom Authentifizierungsprozess der aktuellen Anforderung zugeordnet.

υ So testen Sie die Anwendung

1. Klicken Sie im Menü **Erstellen** auf **Projektmappe erstellen**.
2. Klicken Sie im Projektmappen-Explorer mit der rechten Maustaste auf **default.aspx**, und klicken Sie dann auf **In Browser anzeigen**.
3. Geben Sie einen gültigen Domänennamen, Benutzernamen und das Kennwort ein, und klicken Sie dann auf **Log On**.
4. Wenn Sie erfolgreich authentifiziert werden, sollten Sie auf die Seite **default.aspx** zurückgeführt werden. Der Code auf dieser Seite sollte den Benutzernamen des authentifizierten Benutzers anzeigen.

   Zum Anzeigen der Liste der Gruppen, denen der authentifizierte Benutzer angehört, fügen Sie am Ende des Ereignishandlers **Application_AuthenticateRequest** in der Datei **global.aspx.cs** folgenden Code hinzu.

```
Response.Write("Groups: " + authTicket.UserData + "<br>");
```