CARL HANSER VERLAG

Patrick A. Lorenz

ASP.NET

3-446-22129-8

www.hanser.de

9 Sicherheit

ASP.NET bietet integrierte Sicherheitsmechanismen. Diese nehmen Ihnen einiges an Arbeit ab und ermöglichen so die einfache Erstellung von geschützten Bereichen. Die Mechanismen von .NET arbeiten sehr ähnlich wie die des Menschen. Zunächst wird das Gegenüber identifiziert. Nun da man weiß, wer der Besucher ist, kann auf Basis seiner Person oder aber der zugeordneten Rolle(n) entschieden werden, auf welche Informationen er Zugriff erhält. Der Besucher wird autorisiert oder aber eben nicht. ASP.NET unterteilt in genau diese Stufen, genannt Authentication und Authorization.

9.1 Basics

9.1.1 Authentication

Authentication ist der Prozess des Erkennens. Im Internet sehr verbreitet und ein Quasi-Standard ist die Identifikation eines Besuchers mittels Angabe von persönlichem Benutzernamen und Kennwort, genannt „Credentials“. Auch bei ASP.NET wird dieses grundlegende System eingesetzt. Die Informationen können dabei über verschiedene Authentication Provider abgefragt werden. Die Tabelle zeigt diese in der Übersicht.

Tabelle 9.1 Die ASP.NET Authentication Provider in der Übersicht

Authentication Provider	Beschreibung
Windows Authentication	Bei diesem Verfahren wird die Identifizierung des Besuchers den Internet Information Services (IIS) überlassen. Der Benutzer erhält einen Dialog zur Eingabe seiner Credentials. Diese müssen mit den Daten eines lokal oder in der angegebenen NT-Domäne vorhandenen Benutzers übereinstimmen.

Authentication Provider	Beschreibung
Forms Authentication	Dieses System wird von ASP.NET autark angeboten. Beim ersten Aufruf einer geschützten Seite wird der Besucher auf ein Login-Formular umgeleitet. Nach der Eingabe seiner gültigen Credentials wird ein zeitlich limitiertes, so genanntes Ticket erzeugt, das dem Client als Cookie übermittelt wird. Über dieses Cookie kann der Benutzer anschließend auf die geschützten Seiten der Website zugreifen.
Passport Authentication	Microsoft Passport ist ein zentrales Element der .NET My Services-Initiative des Konzerns aus Redmont. Die Benutzer registrieren sich einmal bei Microsoft und können sich anschließend mit ihren Credentials bei allen angeschlossenen Websites anmelden.
Anonym	Standardmäßig sind die ASP.NET-Sicherheitsmechanismen ausgeschaltet. Der Zugriff auf die Seiten erfolgt daher unter dem in der Datei machines.config angegebenen Account. Standardmäßig ist dies der Benutzer „ASPNET“.

Der gewünschte Authentication Provider wird in der Konfigurationsdatei `web.config` festgelegt. Beachten Sie, dass es sich um die Datei im Hauptverzeichnis der Web-Applikation handeln muss. Eine individuelle Einstellung in den einzelnen Unterverzeichnissen ist nicht möglich.

Im Folgenden finden Sie eine beispielhafte Konfigurationsdatei. Über das Attribut `mode` wird der gewünschte Modus angegeben. Mögliche Werte sind: `Windows`, `Forms`, `Passport` und `None`.

Listing 9.1 web.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
 <system.web>

  <authentication mode="Mode">
  </authentication>

 </system.web>
</configuration>
```

9.1.2 Authorization

Nachdem ein Besucher identifiziert wurde, wird ermittelt, ob er das Recht hat, die angeforderte Ressource in der gewünschten Form abzurufen beziehungsweise auszuführen. An dieser Stelle kommt eine Reihe von unterschiedlichen Sicherheitsstufen zum Einsatz. Zunächst prüfen beispielsweise die Internet Information Services den Zugriff. Nun wird die Anfrage an ASP.NET weitergeleitet, das ebenfalls den Zugriff auf die Datei prüft. Diese Prüfung nennt sich „URL Authorization“, da sie auf Basis der angeforderten URL vorgenommen wird.

An letzter Position steht das Betriebssystem, das unter Windows 2000 die individuelle Freigabe von Ordnern und Dateien auf Benutzer- oder Gruppenebene überprüft. Die Basis bildet hier eine so genannte Access Control List (ACL). Diese Art der Autorisierung nennt sich „File Authorization“, da sie unabhängig von der aufrufenden Applikation (den IIS) auf Dateiebene vorgenommen wird.

9.1.3 Impersonation

Standardmäßig arbeiten die IIS unter dem in der Konfiguration festgelegten Benutzer-Account „ASPNET“. Auf Dateiebene gelten daher die für diesen Benutzer festgelegten Rechte. Eine File Authorization kann somit bei anonymen Benutzern nicht eingesetzt werden, die Authorization wird zur Gänze der URL Authorization überlassen. Ist diese einmal von einem Hacker überwunden, so stehen alle Rechte des konfigurierten Accounts zur Verfügung.

Um mit File Authorization zu arbeiten und die auf Dateiebene vergebenen Benutzerrechte nutzen zu können, bietet ASP.NET die Möglichkeit, die Identität dieses Benutzers anzunehmen. Gegenüber dem Betriebssystem präsentiert sich ASP.NET in diesem Fall unter dem Namen des angegebenen Benutzers. Diese Imitation nennt sich Impersonation.

9.2 Windows Authentication

Die Windows Authentication wird von den Internet Information Services durchgeführt. Voraussetzung hierfür ist, dass der anonyme Zugriff ausgeschaltet ist. Die Benutzerdaten entspringen dem Betriebssystem. Für jeden Benutzer muss somit ein korrespondierender Windows-Account existieren. Auch das Passwort wird auf diese Weise überprüft.

9.2.1 Konfiguration

Um die Windows Authentication nutzen zu können, müssen Sie diese zunächst in der Konfiguration der IIS aktivieren. Sie finden den entsprechenden Dialog auf dem Registerreiter „Verzeichnissicherheit“ der entsprechenden Web-Applikation. Sie können hier aus verschiedenen Sicherheitsmechanismen wählen.

Im zweiten Schritt erfolgt die Konfiguration der Web-Applikation. Das ist leicht, denn es muss lediglich der Authentication-Modus `Windows` gewählt werden. Das Listing zeigt eine vorbereitete `web.config`. Beachten Sie bitte noch einmal den Hinweis, dass Sie diese Konfiguration ausschließlich im Hauptverzeichnis einer Web-Applikation vornehmen können.

Listing 9.2 web.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
 <system.web>

  <authentication mode="Windows">
  </authentication>

 </system.web>
</configuration>
```

9.2.2 Windows Authentication im Einsatz

Nachdem die Konfiguration erfolgreich abgeschlossen ist, wird Ihre Web-Applikation durch die Windows Authentication geschützt. Sobald ein Besucher auf eine Ressource der Applikation zugreifen möchte, wird er nach seinen persönlichen Credentials gefragt. Erst nach Eingabe der gültigen Daten wird der Zugriff gewährt.

Die Anmeldeinformationen des Benutzers werden an die ASP.NET-Engine weitergeleitet. Sie als Entwickler erhalten über die Eigenschaft `User` der Klasse `Page` Zugriff auf diese Daten. Die Eigenschaft liefert eine Klasse, die die Schnittstelle `IPrincipal` unterstützt. Welche Klasse dies tatsächlich ist, hängt von der gewählten Authentifizierung ab. Im vorliegenden Fall der Windows Authentication wird ein Objekt vom Typ `WindowsPrincipal` geliefert.

Tabelle 9.2 Die Mitglieder der Klasse WindowsPrincipal

Mitglied	Rückgabewert	Beschreibung
E Identity	IIdentity	Liefert die zugehörige Identity-Klasse mit weiteren Benutzerdaten
M IsInRole	bool	Gibt an, ob der Benutzer Mitglied einer bestimmten Benutzergruppe ist. Bei der Windows Authentication werden die Gruppen des Betriebssystems ausgewertet.

Über die einzige Eigenschaft `Identity` wird ein Objekt der Klasse `WindowsIdentity` zurückgegeben, das weitergehende Informationen über den angemeldeten Benutzer bereithält, wie beispielsweise dessen Namen. Da als Rückgabewert der Eigenschaft die Schnittstelle `IIdentity` angegeben ist, muss vor dem Zugriff auf die spezifischen Klassenmitglieder unbedingt eine Typenkonvertierung stattfinden.

```
WindowsIdentity Identity = (WindowsIdentity) User.Identity;
```

Da die Klasse nicht in einem der Standard-Namespaces enthalten ist, muss zudem folgende Direktive integriert werden:

```
<% @Import Namespace="System.Security.Principal" %>
```

Das nachfolgende Listing sollte innerhalb einer über Windows Authentication geschützten Web-Applikation abgelegt werden. Die ASP.NET-Seite prüft innerhalb des Page_Load-Ereignisses zunächst ab, ob der Benutzer überhaupt authentifiziert wurde. Hierzu muss die Methode `IsAuthenticated` den booleschen Wert `true` zurückliefern. Ist dies der Fall, wird eine Reihe von Label-Controls an die Eigenschaften der Klasse `WindowsIdentity` gebunden. Ansonsten wird ein negativer Text ausgegeben. Die Umschaltung erfolgt über zwei Panel-Controls.

Listing 9.3 default.aspx

```
<script language="C#" runat=server>

WindowsIdentity Identity;

void Page_Load(object sender, EventArgs e)
{
 Identity = (WindowsIdentity) User.Identity;
```

```
 if(Identity.IsAuthenticated)
  DataBind();
 else
 {
  Panel1.Visible = false;
  Panel2.Visible = true;
 }
}

</script>

<form runat="server">

<ASP:Panel id="Panel1" runat="server">

<h1> Hallo
 <ASP:Label runat="server" Text="<%# Identity.Name %>"/>
</h1>

<p> Sie wurden erfolgreich über
"<ASP:Label runat="server" Text="<%# Identity.AuthenticationType%>"/>"
identifiziert! </p>

<p>
 IsAnonymous: <ASP:Label runat="server"
  Text="<%# Identity.IsAnonymous.ToString() %>"/> <br>
 IsGuest: <ASP:Label runat="server"
  Text="<%# Identity.IsGuest.ToString() %>"/> <br>
 IsSystem: <ASP:Label runat="server"
  Text="<%# Identity.IsSystem.ToString() %>"/>
</p>

</ASP:Panel>

<ASP:Panel id="Panel2" runat="server"
 visible="false">

<p> Leider konnten Sie nicht identifiziert werden ... </p>

</ASP:Panel>

</form>
```

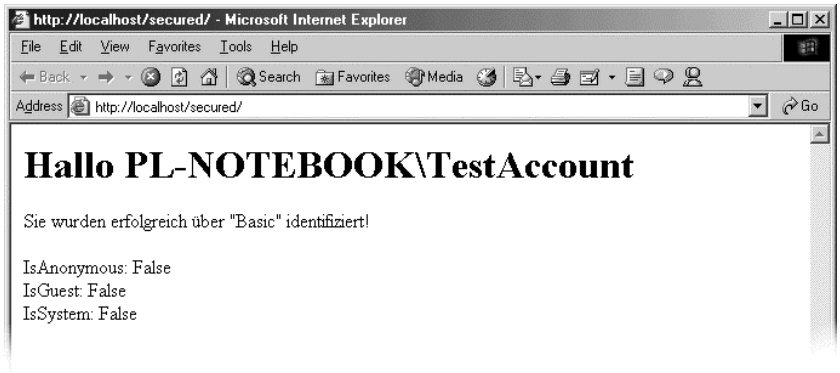

Abbildung 9.1 Der Besucher wurde erfolgreich authentifiziert.

9.2.3 Gruppenmitgliedschaft überprüfen

Sie können die Methode `IsInRole` der Klasse `WindowsPrincipal` dazu verwenden, die Mitgliedschaft eines Benutzers in einer angegebenen Windows-Benutzergruppe zu überprüfen. Dies kann beispielsweise nützlich sein, wenn bestimmte Seitenelemente nur den Administratoren angezeigt werden sollen.

Um beispielsweise abzufragen, ob ein Benutzer Mitglied der Administratoren-Gruppe ist, verwenden Sie nachfolgende Zeilen. Beachten Sie, dass vor der Verwendung der Überladung zunächst eine Typenkonvertierung notwendig ist.

```
WindowsPrincipal Principal = (WindowsPrincipal) User;
bool IsInRole = Principal.IsInRole
 (WindowsBuiltInRole.Administrator);
```

9.2.4 Impersonation

Ist die Impersonation aktiviert, imitiert ASP.NET gegenüber dem Betriebssystem einen bestimmten Benutzer und handelt quasi in dessen Auftrag. Entsprechend gelten auch die Rechte dieses Benutzers. ASP.NET kann somit auf alle freigegebenen Ressourcen zugreifen. Sind bestimmte Dateien hingegen nicht zugänglich, so kann auch ASP.NET diese nicht abrufen.

Automatische Impersonation

Sie können ASP.NET die automatische Verwendung von Impersonation überlassen. Sie müssen dazu lediglich ein entsprechendes Element in die Konfigurationsdatei `web.config` aufnehmen, das die Impersonation aktiviert; standardmäßig ist sie deaktiviert.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
 <system.web>

  <identity impersonate="true"/>

 </system.web>
</configuration>
```

Nun kümmert sich ASP.NET selbstständig um alles weitere und übernimmt den Benutzer-Token, der von den Internet Information Services im Zuge der Client-Anfrage übergeben wird. Sofern der anonyme Zugriff erlaubt ist, wird ein Token auf den dafür konfigurierten Account „ASPNET“ übergeben.

Statische Impersonation

Alternativ können Sie die statische Impersonation verwenden. Dabei geben Sie in der Konfigurationsdatei fest einen Benutzernamen samt zugehörigem Passwort an. Alle Aufrufe werden nun grundsätzlich und unabhängig von der Authentifizierung unter diesem Benutzer-Account durchgeführt.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
 <system.web>

  <identity
   impersonate="true"
   userName="Administrator"
   password="sagichnicht"/>

 </system.web>
</configuration>
```

Dynamische Impersonation

Weiter oben habe ich Ihnen die Klasse `WindowsIdentity` vorgestellt. Das Objekt repräsentiert eine Benutzeridentität. Eine Instanz dieser Klasse erhalten Sie über die Eigenschaften `Page.User.Identity`. Es handelt sich um den Benutzer, der im Zuge der Windows Authentication identifiziert wurde. Über die Methode `Impersonate` haben Sie die Möglichkeit, ab sofort dessen „Gestalt" anzunehmen und unter diesem Account zu agieren.

Die Methode erwartet keinerlei Parameter und liefert ein Objekt vom Datentyp `WindowsImpersonationContext` zurück. Diese Klasse verfügt über eine einzige Methode `Undo`, über die die Impersonation rückgängig gemacht werden kann. Anschließend ist wieder der ursprüngliche Benutzer aktiv.

Im Listing sehen Sie die Verwendung der Methode. Die ASP.NET-Seite gibt innerhalb eines Label-Controls den aktuellen Benutzernamen aus. Dieser wird über die statische Methode `GetCurrent` der Klasse `WindowsIdentity` ermittelt. Anschließend erfolgt die Impersonation, und erneut wird der aktuelle Benutzer ausgegeben. Zu guter Letzt wird die Methode `Undo` aufgerufen und noch einmal der nun aktive Account abgefragt.

Listing 9.4 default.aspx

```
<% @Import Namespace="System.Security.Principal" %>

<script language="C#" runat=server>

void Page_Load(object sender, EventArgs e)
{

 WindowsIdentity Identity;

 // Aktuelle Identität abfragen
 Identity = WindowsIdentity.GetCurrent();
 lb_identity_1.Text = Identity.Name;

 // Impersonation entspricht Windows Authentication
 WindowsIdentity RequestIdentity = (WindowsIdentity) User.Identity;
 WindowsImpersonationContext ImpersonationContext =
  RequestIdentity.Impersonate();

 // Aktuelle Identität abfragen
 Identity = WindowsIdentity.GetCurrent();
 lb_identity_2.Text = Identity.Name;

 // Impersonation zurücknehmen
 ImpersonationContext.Undo();
```