L'authentification par formulaire en ASP.NET

Sommaire

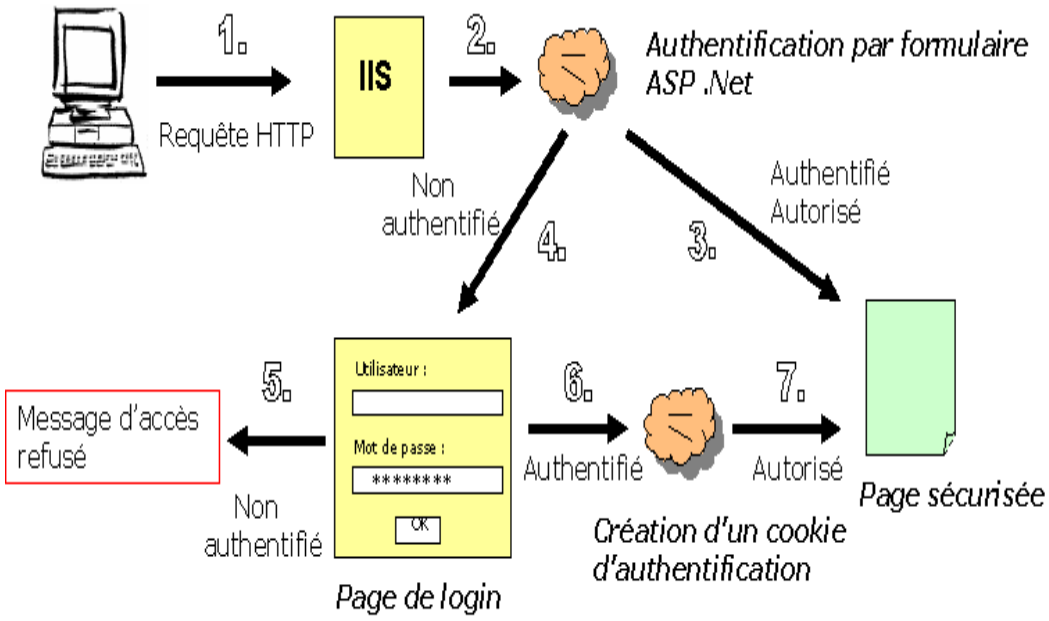
Introduction

Nous allons aborder l'authentification par formulaire. Ce mode d'authentification consiste à rediriger tout utilisateur vers une page de login tant que celui-ci ne s'est pas authentifié. L'authentification peut-être réalisée soit à partir d'un fichier de configuration XML soit à partir d'un système externe (SGBD ou Active Directory).

Nous allons décrire la mise en ouvre du mécanisme d'authentification à base de formulaire. La liste des utilisateurs autorisés sera dans un premier temps décrite dans un fichier de configuration XML puis intégrée à une table des utilisateurs d'une base de données SQL2000.

Regardons plus précisément le fonctionnement interne de l'authentification par formulaire.

1. L'authentification par formulaire sous ASP .Net



Voici comment fonctionne sous ASP .Net l'authentification par formulaire :

1. Un client soumet une requête http pour accéder à une page ASP sécurisée.
2. IIS transmet la requête à ASP .Net pour authentification. Il faut cocher l'accès anonyme pour l'authentification IIS.
3. ASP.NET vérifie si le client dispose d'un cookie d'authentification. Si l'utilisateur n'est pas authentifié alors il est redirigé vers une page de login.
4. L'utilisateur saisit son identité et son mot de passe.
5. Les informations d'authentification sont vérifiées (fichier de configuration, SGBD .). Si l'utilisateur n'est pas authentifié alors un message d'accès refusé est affiché.
6. Les informations d'authentification ont été validées, un cookie d'authentification est généré.
7. Si l'utilisateur est autorisé par ASP.NET alors il accède à la page demandée.

Passons à la pratique et mettons en place ce système d'authentification en déroulant les étapes suivantes :

- Paramétrage du fichier Web.Config : Définition du mode d'authentification.Définition de la page de redirection.Définition de la liste des utilisateurs authentifiés avec cryptage de leur mot de passe.
- Création d'un formulaire Web pour recueillir les informations d'identification du client.
- Vérification des informations d'authentification du client à partir d'un fichier de configuration XML.

2. Paramétrage du fichier Web.Config

Pour positionner l'authentification par formulaire, voici les modifications à réaliser dans le fichier Web.Config.

```
<authentication mode="Forms">
<forms loginUrl="login.aspx" timeout="20">
<credentials passwordFormat="MD5">
<user name="christian" password="7FF13585437685
0E9711BD75CE942E07" />
</credentials>
</forms>
</authentication>
<authorization>
<deny users="?" />
</authorization>
```

Tout d'abord le mode d'authentification a été basculé à "**Forms**". La balise **LoginUrl** indique la page vers laquelle l'utilisateur sera systématiquement redirigé tant qu'il ne sera pas authentifié. La balise **timeout** indique la durée en minutes du cookie d'authentification. Les informations concernant les utilisateurs sont définies dans la balise **credentials**. Il faut ensuite définir pour chaque utilisateur un nom et un mot de passe au sein d'une balise user, seuls ces comptes seront habilités à s'authentifier sur notre application.

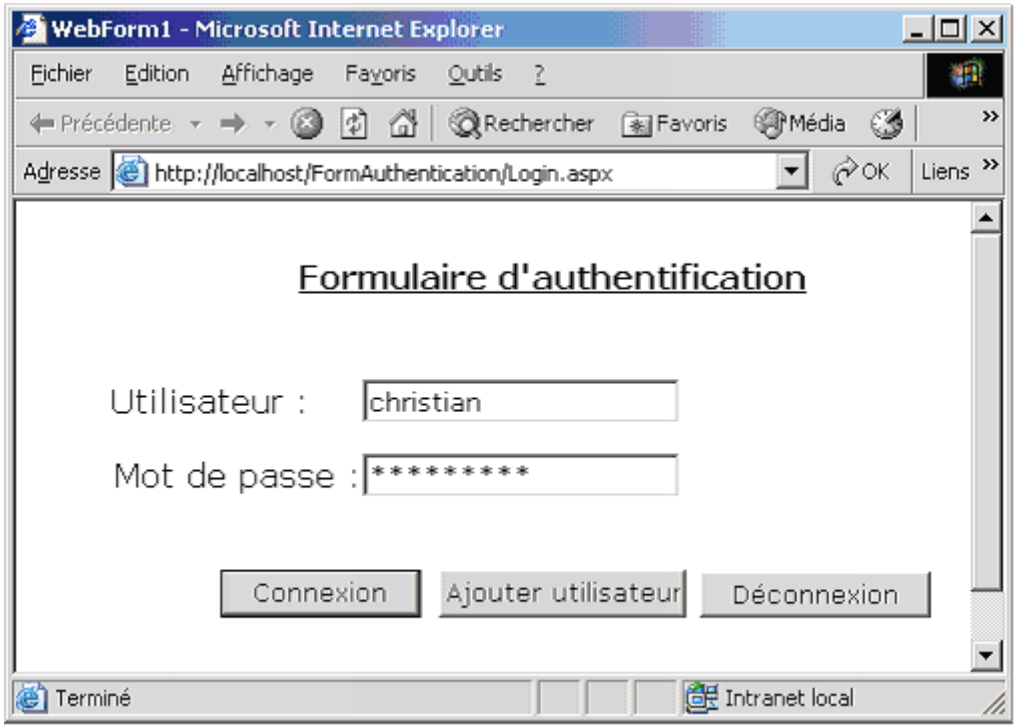
Enfin, nous complétons la sécurité de notre application Web, en interdisant son accès à tous les utilisateurs non authentifiés.

Le cryptage du mot de passe :
Mais me direz-vous que signifie le format MD5 et comment crypter le mot de passe des utilisateurs ?

Le cryptage du mot de passe peut être réalisé grâce à la fonction **HashPasswordForStoringInConfigFile** de la classe **FormsAuthentication**. Elle permet de générer un mot de passe crypté selon un algorithme de hachage SHA1 ou MD5. Ces deux valeurs peuvent être utilisées comme format du mot de passe ainsi qu'une valeur « Clear » à utiliser si vous ne souhaitez pas le crypter.

3. Le formulaire d'authentification

Voici une copie de notre écran d'authentification.



L'écran d'authentification est relativement simple avec une zone de saisie pour le nom de l'utilisateur, une zone de saisie pour le mot de passe et un bouton de connexion. Ne nous préoccupons pas pour l'instant des autres boutons.

Regardons maintenant le code exécuté lorsque l'utilisateur clique sur le bouton Connexion.

```
using System.Web.Security;

....

private void BtConnexion_Click(object sender,
System.EventArgs e)
{
    if (FormsAuthentication.Authenticate(txtUti
lisateur.Text,txtMotDePasse.Text))
    {

FormsAuthentication.RedirectFromLoginPage(txtUt
```

```
ilisateur.Text,false );
     }
     else
     {
          lbMessage.Text = "Login incorrect !";
     }
}
```

La classe FormsAuthentication fait partie de l'espace de noms System.Web.Security. Il faut donc l'inclure dans notre page.

La méthode Authenticate vérifie le nom d'utilisateur et son mot de passe à partir des éléments contenus dans la balise Credentials du fichier Web.Config.. Si l'utilisateur est habilité alors un cookie d'authentification est généré et l'utilisateur est redirigé vers la page demandée initialement ; ces opérations sont réalisées par la méthode RedirectFromLoginPage. Cette méthode prend également un second paramètre qui indique si le cookie doit persister lors d'un changement de session. Précisons que dans le cas où la méthode RedirectFromLoginPage ne peut pas identifier de page de retour alors la page default.aspx est affichée.

4. L'authentification des utilisateurs via une base de données

Nous venons de mettre en place une authentification par formulaires en validant l'identité de l'utilisateur à partir d'informations contenues dans un fichier de configuration XML. Cette solution fonctionne mais peut devenir rapidement fastidieuse pour un administrateur chargé de saisir un grand nombre de comptes utilisateurs.

Essayons d'améliorer notre solution en validant désormais les informations à partir d'une table "utilisateurs" d'une base de données SQL2000. Pour ce faire, nous devons :

- Modifier le fichier Web.Config en supprimant la balise **credentials**.
- Créer une table SQL2000 qui contiendra les informations d'authentification.
- Développer notre propre fonction d'authentification "Authentifier" chargée de vérifier les informations saisies par l'utilisateur avec celles contenues en base.
- Modifier le code associé au bouton Connexion afin d'utiliser notre nouvelle méthode.

Description de la table "utilisateurs" :

Champ	Type de données SQL2000	Description
nom	Varchar(20)	Nom de l'utilisateur
mot de passe	Varchar(50)	Mot de passe

Pour des raisons de sécurité, nous crypterons le mot de passe au format MD5.

Description de la fonction "Authentifier" :

```
private bool Authentifier(string strUtilisateur
, string strMotDePasse)
{
    bool bOk=false ;
    // Cryptage du mot de passe
    strMotDePasse =
FormsAuthentication.HashPasswordForStoringInCon
figFile(strMotDePasse,"MD5");
    // Création d'une connexion SGBD
    SqlConnection oConnexion
= new SqlConnection("user
id=sa;password=;initial catalog=pubs;data
source=pttravail");
    // Définition de la requête à exécuter
    SqlCommand oCommand
= new SqlCommand("SELECT * FROM Utilisateurs
WHERE nom='" + strUtilisateur+ "'",oConnexion);
    try
    {
        // Ouverture de la connexion et
exécution de la requête
        oConnexion.Open();
        SqlDataReader drUtilisateur =
oCommand.ExecuteReader();
```

```
        // Parcours de la liste des
utilisateurs
        while (drUtilisateur.Read())
        {

    if (drUtilisateur["motdepasse"].ToString()
== strMotDePasse)
            {
                bOk = true ; break ;
            }
        }
    }
    catch
    {
        bOk = false ;
    }
    oConnexion.Close();
    return bOk;
}
```

Nous cryptons d'abord le mot de passe saisis par l'utilisateur, puis nous sélectionnons la liste des mots de passe de l'utilisateur qui cherche à s'authentifier, enfin nous la parcourons en comparant chaque mot de passe avec celui crypté précédemment. Dès qu'un mot de passe a été trouvé nous retournons vrai à la fonction appelante.

Il ne nous reste plus qu'a modifier le code associé au bouton Connexion et à invoquer notre méthode Authentifier.

```
private void BtConnexion_Click(object sender,
System.EventArgs e)
{
    if (Authentifier(txtUtilisateur.Text,txtMot
DePasse.Text))
    {

FormsAuthentication.RedirectFromLoginPage(txtUt
ilisateur.Text,false );
    }
    else
    {
        lbMessage.Text = "Erreur
d'authentification, l'utilisateur ou le mot de
passe n'existent pas!";
```

```
        }
}
```

5. La déconnexion de l'utilisateur

La plupart des sites, qui proposent un moyen de s'authentifier par formulaires, fournissent également un bouton de déconnexion. La méthode SignOut() de la classe FormsAuthentication réalise cette opération de déconnexion en détruisant le cookie d'authentification, ce qui obligera l'utilisateur à s'authentifier de nouveau pour accéder à l'une des pages de votre application.

Voici le code exécuté lorsque l'utilisateur clique sur le bouton Déconnexion :

```
private void BtDeconnexion_Click(object sender,
System.EventArgs e)
{
    FormsAuthentication.SignOut();
Response.Redirect("Login.aspx");
}
```

Nous détruisons le cookie d'authentification et redirigeons l'utilisateur vers la page de login.

6. Conclusion

Nous avons pu nous rendre compte de la facilité de mise en oeuvre de l'authentification par formulaire avec ASP .Net. Il est ainsi possible en quelques lignes de code et à moindre coût de sécuriser toute application Web Intranet ou Internet.