Comprendre les services d'authentification et de profil d'ASP.NET AJAX

Sommaire

Introduction

Dans le Framework .NET 3.5, Microsoft fournit une importante mise à niveau de l'environnement ; non seulement un nouvel environnement de développement est disponible, mais LINQ (Language-Integrated Query, Requête intégrée au langage) et d'autres améliorations linguistiques des améliorations sont à venir. En outre, certaines caractéristiques familières d'autres outils, notamment les extensions ASP.NET AJAX, sont désormais incluses en tant que citoyennes de première classe dans la bibliothèque de classes du Framework. Ces extensions permettront de nombreuses nouvelles fonctionnalités client riche, y compris le rendu partiel des pages, sans nécessiter un rafraîchissement de toute la page, la capacité d'accéder à des services Web via des scripts coté client (y compris l'API de profilage d'ASP.NET), et une vaste API coté client, conçue pour refléter de nombreux schémas de contrôle existants dans les contrôles serveurs d'ASP.NET.

Ce livre blanc se penche sur la mise en oeuvre et l'utilisation des services d'authentification et de profils d'ASP.NET tels qu'ils sont exposés par le Framework ASP.NET AJAX Extensions. Les extensions AJAX rendent l'authentification par formulaires incroyablement facile à maintenir, comme il est (ainsi que le service de profils) exposé par le biais d'un script de proxy de service Web. Les AJAX Extensions permettent aussi une authentification personnalisée par l'intermédiaire de la classe AuthenticationServiceManager.

Ce livre blanc est fondé sur la version bêta 2 de Visual Studio 2008 et du Framework .NET 3.5. Ce livre blanc suppose également que vous travaillez avec Visual Studio 2008 Beta 2, et pas avec Visual Web Developer Express, et fournira des solutions basées sur l'interface utilisateur de Visual Studio. Quelques exemples de code pourraient utiliser des modèles de projets non disponibles dans Visual Web Developer Express.

Profils et authentification

Les services d'authentification et de profil d'ASP.NET sont fournis par le système d'authentification par formulaire d'ASP.NET, et sont des composants standard d'ASP.NET. Les extensions ASP.NET AJAX fournissent des accès sous forme de script à ces services via des script proxy, par le biais d'un modèle assez simple dans l'espace de noms Sys.Services de la bibliothèque client AJAX.

Le service d'authentification (Authentication service) permet aux utilisateurs de fournir des références (credentials), en vue de recevoir un cookie d'authentification et est le point d'accès pour permettre l'utilisation de profils d'utilisateur personnalisés, fournis par ASP.NET. L'utilisation du service d'authentification ASP.NET AJAX est compatible avec l'authentification par formulaires standard d'ASP.NET, par conséquent, une mise à niveau vers le service d'authentification AJAX ne provoquera pas de bug dans des applications utilisant l'authentification par formulaires (par exemple avec un contrôle de type Login).

Le service de profil permet l'intégration et le stockage automatique des données de l'utilisateur en fonction de son appartenance à un groupe, comme ceux fournis par le service d'authentification. Les données stockées sont définies par le fichier web.config, et les différents fournisseurs de services de profils prennent en charge la gestion des données. Comme dans le service

d'authentification, le service de profil AJAX est compatible avec le service de profil standard d'ASP.NET, de sorte que les pages intégrant les fonctionnalités du service de profil d'ASP.NET ne seront pas endommagées par le support d'AJAX.

L'intégration des services d'authentification et de profil d'ASP.NET dans une application est hors de la portée de ce livre blanc. Pour plus d'informations sur le sujet, voir l'article de la bibliothèque de référence MSDN "Gestion des utilisateurs à l'aide de l'appartenance", à **http://msdn.microsoft.com/fr-fr/library/tw292whz.aspx**. ASP.NET comprend aussi un utilitaire pour mettre en place automatiquement un gestionnaire d'appartenance pour SQL Server, qui est le fournisseur de services d'authentification par défaut pour l'appartenance ASP.NET. Pour plus d'informations, voir l'article "ASP.NET SQL Server Registration Tool (Aspnet_regsql.exe)" à **http://msdn.microsoft.com/fr-fr/library/ms229862(vs.80).aspx**.

Utilisation du service d'authentification ASP.NET AJAX

Le service d'authentification ASP.NET AJAX doit être activé dans le fichier web.config :

```
<system.web.extensions>
    <scripting>
            <webServices>
                    <authenticationService enabled="true" />
            </webServices>
    </scripting>
</system.web.extensions>
```

Le service d'authentification requiert l'activation de l'authentification par formulaire ASP.NET, ainsi que l'activation des cookies sur le navigateur du client (un script ne permet pas une session sans cookie car une session sans cookie nécessite l'ajout de paramètres dans l'URL).

Une fois que le service d'authentification AJAX est activé et configuré, les scripts clients peuvent immédiatement profiter de l'objet **Sys.Services.AuthenticationService**. Principalement, les scripts clients voudront utiliser la méthode **Login**, ainsi que la propriété **isLoggedIn**. Plusieurs propriétés existent pour fournir des valeurs par défaut pour la méthode **Login**, qui peut accepter un grand nombre de paramètres.

Membres de Sys.Services.AuthenticationService

Méthode login

La méthode login() initie la requête d'authentification des crédits de connexion de l'utilisateur. Cette méthode est asynchrone et ne bloque pas l'exécution.

Paramètres

Nom de paramètre	Signification
userName	Requis. Le nom d'utilisateur à authentifier.
password	(Optionnel, *null* par défaut). Le mot de passe utilisateur.
isPersistent	(Optionnel, *false* par défaut). Détermine si le cookie d'authentification de l'utilisateur doit être persisté à travers les sessions. Si false, l'utilisateur se déconnecte lorsque le navigateur est fermé ou que la session expire.
redirectUrl	(Optionnel, *null* par défaut). L'adresse URL à laquelle rediriger le navigateur en cas de succès durant l'authentification. Si ce paramètre vaut *null*ou une chaîne vide, aucune redirection ne se produit.
customInfo	(Optionnel, *null* par défaut). Ce paramètre est actuellement inutilisé et est réservé pour une utilisation future.
loginCompletedCallback	(Optionnel, *null* par défaut). La fonction a appeler lorsque la connexion a réussi. Si spécifié, ce paramètre prime sur la propriété**defaultLoginCompleted**.
failedCallback	(Optionnel, *null* par défaut). La fonction a appeler lorsque la connexion a échoué. Si spécifié, ce paramètre prime sur la propriété**defaultFailedCallback**.
userContext	(Optionnel, *null* par défaut). Données de contexte utilisateur qui devraient être passées aux fonctions de callback.

Valeur de retour

Cette fonction ne comporte pas de valeur de retour. Toutefois, un certain nombre de comportements sont possibles à l'issue d'un appel à cette fonction :

- La page en cours sera mise à jour ou modifiée si le paramètre **redirectUrl** n'est ni *null* ni une chaîne vide.
- Toutefois, si le paramètre est *null* ou une chaîne vide, la fonction définie dans le paramètre **loginCompletedCallback**, ou dans la propriété **defaultLoginCompletedCallback** est appelée.
- Si l'appel au service Web échoue, la fonction définie dans le paramètre **failedCallback** ou dans la propriété **defaultFailedCallback** est appelée.

Méthode logout

La méthode logout() supprime les cookie de connexion et déconnecte l'utilisateur courant de l'application Web.

Paramètres

Nom de paramètre	Signification
redirectUrl	(Optionnel, *null* par défaut). L'adresse URL vers laquelle rediriger le navigateur lors d'une authentification réussie. Si ce paramètre est *null* ou une chaîne vide, aucune redirection ne se produit.
logoutCompletedCallback	(Optionnel, *null* par défaut). Fonction appelée lorsque la déconnexion a réussi. Si spécifié, ce paramètre prime sur la propriété**defaultLogoutCompleted**.
failedCallback	(Optionnel, *null* par défaut). Fonction appelée lorsque la connexion a échoué. Si spécifié, ce paramètre prime sur la propriété**defaultFailedCallback**.
userContext	(Optionnel, *null* par défaut). Données de contexte utilisateur qui devraient être passées aux fonctions de callback.

Valeur de retour

Cette fonction ne comporte pas de valeur de retour. Toutefois, un certain nombre de comportements sont possibles à l'issue d'un appel à cette fonction :

- La page en cours sera mise à jour ou modifiée si le paramètre **redirectUrl** n'est ni *null* ni une chaîne vide.

- Toutefois, si le paramètre est *null* ou une chaîne vide, la fonction définie dans le paramètre **logoutCompletedCallback**, ou dans la propriété **defaultLogoutCompletedCallback** est appelée.
- Si l'appel au service Web échoue, la fonction définie dans le paramètre **failedCallback** ou dans la propriété **defaultFailedCallback** est appelée.

Propriété defaultFailedCallback (get, set)

Cette propriété spécifie une fonction qui doit être invoquée si une erreur de communication avec le service web se produit. Elle devrait recevoir un délégué (ou une référence à une fonction).

La fonction spécifiée par cette propriété doit avoir la signature suivante :

```
function AuthenticationFailureCallback(error, userContext,
methodName);
```

Paramètres

Nom de paramètre	Signification
error	Spécifie les informations d'erreur.
userContext	Données de contexte utilisateur qui devraient être passées aux fonctions de callback.
methodName	Le nom de la méthode à appeler.

Propriété defaultLoginCompletedCallback (get, set)

Cette propriété spécifie une fonction qui doit être invoquée lorsque l'appel au service web **login** s'est effectué avec succès. Elle devrait recevoir un délégué (ou une référence à une fonction).

La fonction spécifiée par cette propriété doit avoir la signature suivante :

```
function
AuthenticationLoginCompletedCallback(validCredentials,
userContext, methodName);
```

Paramètres

Nom de paramètre	Signification
validCredentials	Indique si l'utilisateur a fourni des crédits de connexion valides. **true** si l'utilisateur à réussi à se connecter, **false** sinon.
userContext	Données de contexte utilisateur qui devraient être passées aux fonctions de callback.
methodName	Le nom de la méthode à appeler.

Propriété defaultLogoutCompletedCallback (get, set)

Cette propriété spécifie une fonction qui doit être invoquée lorsque l'appel au service web **logout** s'est effectué avec succès. Elle devrait recevoir un délégué (ou une référence à une fonction).

La fonction spécifiée par cette propriété doit avoir la signature suivante :

```
LogoutCompletedCallback(result, userContext, methodName);
```

Paramètres

Nom de paramètre	Signification
result	Ce paramètre sera toujours **null**, il est réservé pour une utilisation future.
userContext	Données de contexte utilisateur qui devraient être passées aux fonctions de callback.
methodName	Le nom de la méthode à appeler.

Propriété isLoggedIn (get)

Cette propriété récupère l'état actuel d'authentification de l'utilisateur; il est fixé par l'objet ScriptManager lors d'une requête.

Cette propriété renvoie **true** si l'utilisateur est actuellement connecté, sinon, elle retourne **false**

Propriété path (get, set)

Cette propriété détermine programmatiquement l'emplacement du service web d'authentification. Il peut être utilisé pour surcharger le fournisseur d'authentification par défaut, de la même façon qu'en utilisant la propriété du noeud enfant AuthenticationService du contrôle ScriptManager (pour plus d'informations, consultez la section "Utilisation d'un fournisseur de service d'authentification personnalisé" ci-dessous).

Notez que l'emplacement du service d'authentification par défaut ne change pas. Cependant, ASP.NET AJAX vous permet de spécifier l'emplacement d'un service Web qui fournit la même interface de classe que le proxy du service d'authentification ASP.NET AJAX.

Notez aussi que cette propriété ne doit pas être fixée à une valeur qui dirige le script en dehors du site courant. En effet, l'application courante ne recevrait pas l'authentification, ce qui serait inutile; de plus, la technologie sous-jacente à AJAX ne doit pas poster de requête vers un autre site, et pourrait générer une exception de sécurité dans le navigateur du client.

Cette propriété est un objet **String** représentant le chemin d'accès au service web d'authentification.

Propriété timeout (get, set)

Cette propriété détermine la durée des temps d'attente pour le service d'authentification avant de considérer que la demande de connexion a échoué. Si le délai d'expiration intervient alors que l'on attend qu'un appel au service soit complété, le caalback **request-failed** ne sera pas appelé, et l'appel ne sera pas complété.

Cette propriété est un objet **Number**, qui représente le nombre de millisecondes que le script passera à attendre les résultats du service d'authentification.

Exemple de code: connexion au service d'authentification

Le code suivant est un exemple de page ASP.NET avec un script simple faisant appel aux méthodes **login** et **logout** de la classe **AuthenticationService**.

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

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head runat="server">
            <title>Login Example</title>
            <script type="text/javascript">

                    function Login(){
                            var userTextbox = $get("txtUser");
                            var passTextbox = $get("txtPassword");

    Sys.Services.AuthenticationService.login(userTextbox.value,
                                    passTextbox.value, false, null,
null, LoginServiceCompleted,
                                    LoginServiceFailed, "Context
Info");
                    }

                    function Logout(){

    Sys.Services.AuthenticationService.logout(null,
LogoutServiceCompleted,
                                    LoginServiceFailed, "Context
Info");
                    }

                    function LoginServiceFailed(error, userContext,
methodName){
                            alert('There was an error with the
authentication service:\n\n' + error);
                    }

                    function
LoginServiceCompleted(validCredentials, userContext, methodName){
                            if (validCredentials){
                                    alert('Great! You successfully
logged in.');
                            }else{
                                    alert('Oops! You gave us bad
credentials!');
                            }
                    }

                    function LogoutServiceCompleted(result,
userContext, methodName){
                            alert('You have been logged out from the
web site.');
                    }

            </script>
    </head>
    <body>
            <form id="form1" runat="server">
            <asp:ScriptManager ID="ScriptManager1" runat="server"
                    EnableScriptLocalization="true">
            </asp:ScriptManager>
            <div>
                    <asp:TextBox ID="txtUser"
runat="Server"></asp:TextBox>
                    <br />
                    <asp:TextBox ID="txtPassword" runat="Server"
```

```
    TextMode="Password"/>
                    <br />
                    <asp:Button Text="Log in" ID="btnLogin"
    runat="server"
                            OnClientClick="Login(); return false;"
    />
            </div>
            </form>
        </body>
    </html>
```

Accéder aux données du service de profil ASP.NET par AJAX

Le service de profil ASP.NET est également exposé à travers les extensions ASP.NET AJAX. Comme service de profil ASP.NET fournit une API riche et granulaire, qui permet de stocker et de récupérer les données de l'utilisateur, cela peut être un excellent outil de productivité.

Le profil de service doit être activé dans le web.config, ce qu'il n'est pas par défaut. To do so, ensure that the profileService child element has enabled=”true” specified in web.config, and that you have specified which properties can be read or written as follows: Pour ce faire, veillez à ce que l'élément **profileService** ait la propriété **enabled** avec la valeur "true" dans le web.config, et que vous avez spécifié quelles propriétés peuvent être lues ou écrites de cette façon :

```
    <system.web.extensions>
        <scripting>
                <webServices>
                        <profileService enabled="true"

        readAccessProperties=”Name,Address,BackgroundColor”
                                writeAccessProperties=”BackgroundColor”/>
                </webServices>
        </scripting>
    </system.web.extensions>
```

Le service de profil doit aussi être configuré. Bien que la configuration du service de profil soit en dehors de la portée de ce livre blanc, il est intéressant de noter que des propriétés définies dans les paramètres de configuration d'un profil seront accessibles en tant que sous-propriétés du groupe. Par exemple, avec la section profil suivante spécifiée dans le web.config :

```
     <profile enabled="true">
        <properties>
                <add name="Name" type="System.String"/>
                <group name="Address">
                        <add name="Line1" type="System.String"/>
                        <add name="Line2" type="System.String"/>
                        <add name="City" type="System.String"/>
                        <add name="State" type="System.String"/>
                        <add name="Zip" type="System.String"/>
                </group>
                <add name="BackgroundColor"
    type="System.Drawing.Color"/>
```

```
        </properties>
    </profile>
```

Les scripts coté client seraient en mesure d'accéder à Name, Address.Line1, Address.Line2, Address.City, Address.State, Address.Zip, et BackgroundColor en tant que propriétés du champ **properties** de la classe **ProfileService**.

Membres de Sys.Services.ProfileService

Champ properties

The properties field exposes all configured profile data as child properties that can be referenced by the dot-operator-name convention. Le champ **properties** expose toutes les données de profil configurées en tant que propriétés descendantes (child properties), qui peuvent ainsi être référencé par la convention point-opérateur-nom. Les propriétés qui sont les descendantes de groupes de propriétés sont désignées comme GroupName.PropertyName. Dans l'exemple de profil de configuration présenté ci-dessus, pour obtenir l'état (au sens état de résidence) de l'utilisateur, vous pouvez utiliser l'identifiant suivant :

```
Sys.Services.ProfileService.properties.Address.State
```

Méthode load

Charge une liste sélectionnée ou toutes les propriétés depuis le serveur.

Paramètres

Nom de paramètre	Signification
propertyNames	(Optionnel, *null* par défaut). Les propriétés à charger depuis le serveur.
loadCompletedCallback	(Optionnel, *null* par défaut). La fonction à appeler lorsque le chargement a réussi. Si spécifié, ce paramètre prime sur la propriété**defaultLoadCompletedCallback** .
failedCallback	(Optionnel, *null* par défaut). La fonction à appeler lorsque la connexion a échoué. Si spécifié, ce paramètre prime sur la propriété**defaultFailedCallback**.

userContext	(Optionnel, *null* par défaut). Données de contexte utilisateur qui devraient être passées aux fonctions de callback.

Valeur de retour

Cette fonction ne comporte pas de valeur de retour. Si l'appel est effectué avec succès, la fonction définie dans le paramètre **loadCompletedCallback**, ou dans la propriété **defaultLoadCompletedCallback**sera appelée. Si l'appel a échoué, ou que le délai d'expiration arrive à son terme, la fonction définie dans le paramètre **failedCallback** ou dans la propriété **defaultFailedCallback** sera appelée.

Si le paramètre **propertyNames** n'est pas fourni, toutes les propriétés configurées en lecture sont extraites du serveur.

Méthode save

La méthode save() enregistre la liste des propriétés spécifiées (ou de toutes les propriétés) dans le profil ASP.NET de l'utilisateur.

Paramètres

Nom de paramètre	Signification
propertyNames	(Optionnel, *null* par défaut). Les propriétés à sauvegarder sur le serveur.
saveCompletedCallback	(Optionnel, *null* par défaut). La fonction à appeler lorsque la sauvegarde a réussi. Si spécifié, ce paramètre prime sur la propriété**defaultSaveCompletedCallback**.
failedCallback	(Optionnel, *null* par défaut). La fonction à appeler lorsque la connexion a échoué. Si spécifié, ce paramètre prime sur la propriété**defaultFailedCallback**.
userContext	(Optionnel, *null* par défaut). Données de contexte utilisateur qui devraient être passées aux fonctions de callback.

Valeur de retour

Cette fonction ne comporte pas de valeur de retour. Si l'appel est effectué avec succès, la fonction définie dans le paramètre **saveCompletedCallback**, ou dans la propriété**defaultSaveCompletedCallback** sera appelée. Si l'appel a échoué, ou que le délai d'expiration arrive à son terme, la fonction définie dans le paramètre **failedCallback** ou dans la propriété**defaultFailedCallback** sera appelée.

Si le paramètre **propertyNames** n'est pas fourni, toutes les propriétés du profile seront envoyées au serveur, qui décidera quelles propriétés peuvent être sauvegardés, et quelles propriétés ne peuvent pas l'être.

Propriété defaultFailedCallback (get, set)

Cette propriété spécifie une fonction qui doit être invoquée si une erreur de communication avec le service web se produit. Elle devrait recevoir un délégué (ou une référence à une fonction).

La fonction spécifiée par cette propriété doit avoir la signature suivante :

```
function AuthenticationFailureCallback(error, userContext, methodName);
```

Paramètres

Nom de paramètre	Signification
error	Spécifie les informations d'erreur.
userContext	Données de contexte utilisateur qui devraient être passées aux fonctions de callback.
methodName	Le nom de la méthode à appeler.

Propriété defaultSaveCompleted (get, set)

Cette propriété spécifie une fonction qui doit être invoquée lorsque la sauvegarde des données de l'utilisateur s'est effectuée avec succès. Elle devrait recevoir un délégué (ou une référence à une fonction).

La fonction spécifiée par cette propriété doit avoir la signature suivante :

```
```

```
function ProfileSaveComplete(numPropsSaved, userContext, methodName);
```

Paramètres

Nom de paramètre	Signification
numPropsSaved	Indique le nombre de propriétés qui ont été sauvés.
userContext	Données de contexte utilisateur qui devraient être passées aux fonctions de callback.
methodName	Le nom de la méthode à appeler.

Propriété defaultLoadCompleted (get, set)

Cette propriété spécifie une fonction qui doit être invoquée lorsque le chargement des données de l'utilisateur s'est effectué avec succès. Elle devrait recevoir un délégué (ou une référence à une fonction).

La fonction spécifiée par cette propriété doit avoir la signature suivante :

```
function ProfileLoadComplete(numPropsLoaded, userContext,
methodName);
```

Paramètres

Nom de paramètre	Signification
numPropsLoaded	Indique le nombre de propriétés qui ont été chargées.
userContext	Données de contexte utilisateur qui devraient être passées aux fonctions de callback.
methodName	Le nom de la méthode à appeler.

Propriété path (get, set)

Cette propriété détermine programmatiquement l'emplacement du service web d'authentification. Il peut être utilisé pour surcharger le fournisseur d'authentification par défaut, de la même façon qu'en utilisant la propriété du noeud enfant AuthenticationService du contrôle ScriptManager (pour

plus d'informations, consultez la section "Utilisation d'un fournisseur de service d'authentification personnalisé" ci-dessous).

Notez que l'emplacement du service d'authentification par défaut ne change pas. Cependant, ASP.NET AJAX vous permet de spécifier l'emplacement d'un service Web qui fournit la même interface de classe que le proxy du service d'authentification ASP.NET AJAX.

Notez aussi que cette propriété ne doit pas être fixée à une valeur qui dirige le script en dehors du site courant. En effet, l'application courante ne recevrait pas l'authentification, ce qui serait inutile; de plus, la technologie sous-jacente à AJAX ne doit pas poster de requête vers un autre site, et pourrait générer une exception de sécurité dans le navigateur du client.

Cette propriété est un objet **String** représentant le chemin d'accès au service web d'authentification.

Propriété timeout (get, set)

Cette propriété détermine la durée des temps d'attente pour le service d'authentification avant de considérer que la demande de connexion a échoué. Si le délai d'expiration intervient alors que l'on attend qu'un appel au service soit complété, le calback **request-failed** ne sera pas appelé, et l'appel ne sera pas complété.

Cette propriété est un objet **Number**, qui représente le nombre de millisecondes que le script passera à attendre les résultats du service d'authentification.

Exemple de code : Récupération des données de profil durant le chargement d'une page

Le code suivant vérifie si un utilisateur est authentifié et, le cas échéant, charge depuis les préférences de l'utilisateur la couleur de fond de la page.

```
function Page Load(){
    if (Sys.Services.AuthenticationService.get_isLoggedIn()){
            Sys.Services.ProfileService.load();
    }
}

function ProfileLoaded(numPropsLoaded, userContext, methodName){
    document.documentElement.style.backgroundColor =
Sys.Services.ProfileService.properties.BackgroundColor;
}
```

Note du traducteur : le code de cet exemple me semble incomplet, il n'y à pas en effet de mise en relat

De deux choses l'une, soit il manque, au-dessus de l'appel à **load**, une ligne :

```
Sys.Services.ProfileService.set_defaultLoadCompletedCallback(Profil
eLoaded);
```

Soit la fonction **load** devrait être appelée avec la signature suivante:

```
ys.Services.ProfileService.load(null, ProfileLoaded, null, null);
```

Utilisation d'un fournisseur de service d'authentification personnalisé

Les extensions ASP.NET AJAX vous permettent de créer un fournisseur de service d'authentification personnalisé, en exposant votre fonctionnalité par l'intermédiaire d'un service web personnalisé. Pour être utilisé, votre service web doit exposer deux méthodes, **Login** et **Logout**, et ces méthodes doivent être spécifiées avec la même signature que le service web par défaut d'authentification ASP.NET AJAX.

Une fois que vous avez créé le service Web personnalisé, vous devez spécifier le chemin vers ce service, que ce soit de façon déclarative sur votre page, programmatiquement dans le code, ou via un script coté client.

Définition du chemin de façon déclarative

Pour spécifier le chemin de façon déclarative, il faut ajouter à l'objet ScriptManager défini sur votre page ASP.NET la balise enfant AuthenticationService :

```
ScriptManager1" runat="server">
rvice Path="~/AuthService.asmx" />
```

Définition du chemin dans le code

Pour définir le chemin programmatiquement, indiquez le chemin d'accès via l'instance de votre script manager:

```
protected void Page_Load(object sender, EventArgs e) {
    this.ScriptManager1.AuthenticationService.Path =
"~/AuthService.asmx";
}
```

Définition du chemin dans le script coté client

Pour définir le chemin dans le script coté client, utilisez la propriété **path** de la classe AuthenticationService :

```
function Login() {
    var userTextbox = $get("txtUser");
    var passTextbox = $get("txtPassword");
    Sys.Services.AuthenticationService.set_path("./AuthService.asmx");
    Sys.Services.AuthenticationService.login(userTextbox.value,
passTextbox.value, false,
                null, null, LoginServiceCompleted, LoginServiceFailed,
"Context Info");
}
```

Exemples de services Web pour une authentification personnalisée

```
<%@ WebService Language="C#" Class="AuthService" %>

using System;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;
using System.Web.Script.Services;

[ScriptService]
[WebService]
public class AuthService : WebService {
    [WebMethod]
    public bool Login(string userName, string password, bool
createCookie){
            Session["LoggedInUser"] = userName;
            return true;
    }

    [WebMethod]
    public void Logout(){
            Session.Abandon();
    }
}
```

Conclusion

Les services ASP.NET - en particulier, les services de profil, d'appartenance à un groupe (membership), et d'authentification - sont facilement exposés en JavaScript sur le navigateur du client. Cela permet aux développeurs d'intégrer leur code côté client avec les mécanismes d'authentification de façon transparente, sans dépendre de contrôles tels que les UpdatePanels et de manipulations lourdes. Les données de profil peuvent aussi être protégées du client, en utilisant des paramètres de configuration web; aucune donnée n'est disponible par défaut, et les développeurs doivent activer pour les propriétés de profil.

En outre, en créant des implémentations de services Web simplifiées avec des signatures de méthode équivalentes, les développeurs peuvent créer des fournisseurs de scripts personnalisés pour ces services intrinsèques à ASP.NET. Le support de ces techniques simplifie le développement d'applications client riche, tout en fournissant aux développeurs une grande flexibilité pour répondre à des besoins spécifiques.