Le CASI (Claims, Azure and SharePoint Integration) Kit

Partie 6 - Intégration de SQL Azure avec SharePoint 2010 et Windows Azure

Par Steve Peschka - Deepin Prayag (traducteur)

Date de publication : 21 janvier 2013

Ceci est une série d'articles à propos de laquelle je suis plutôt enthousiaste, et j'espère que tout compte fait vous le serez aussi. Je travaille depuis plusieurs mois sur un nouveau « framework » permettant de connecter SharePoint et Windows Azure, ainsi que d'incorporer l'identité basée sur les revendications d'une manière qui permette à votre identité d'être transmise de façon transparente indépendamment des limites des applications et même des centres de données. L'aboutissement de cet effort - désormais connu sous le nom de CASI (Claims, Azure, and SharePoint Integration) Kit - est une combinaison d'instructions et de lignes directrices, d'assembly de classe de base, de web part et d'exemples d'applications. Ensemble, ces éléments vous permettront de créer des applications WCF qui prennent en charge les revendications et de les héberger dans le cloud Windows Azure.

Langage/Outil/Technologie : SharePoint 2010, Windows Azure, CASI Kit.

Commentez ce tutoriel :

Traduction....3
Introduction....3
Le CASI Kit : Partie 6 - Intégration de SQL Azure avec SharePoint 2010 et Windows Azure....3
Conseils d'intégration....4
Accès aux données....5
Conclusion....7
Liens....7
Remerciements....7

Copyright ® 2013 Steve Peschka. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Traduction

Cet article est la traduction la plus fidèle possible de l'article original de **Steve Peschka**, **The Claims, Azure and SharePoint Integration Toolkit Part 6**.

Introduction

Ceci est la sixième partie d'une série en six parties sur le CASI (Claims, Azure and SharePoint Integration) Kit. Cet article est plus utile lorsqu'il est utilisé en complément des cinq premières parties de la série.

- **Partie 1** : une introduction présentant une vue d'ensemble de l'intégralité du « framework » et de la solution, et qui décrit ce que la série va tâcher de couvrir.
- **Partie 2** : la partie « instructions et lignes directrices » du CASI Kit. Cela commence par faire de WCF le « frontend » pour toutes vos données - cela pourrait être des jeux de données, du XML, des classes personnalisées ou tout simplement de l'HTML. Dans la première phase, nous prenons votre service WCF standard et en faisons un service prenant en charge les revendications - c'est ce qui nous permet de prendre le jeton utilisateur de SharePoint et de le transmettre, indépendamment des limites des applications ou des centres de données, vers nos applications WCF personnalisées. Dans la deuxième phase, j'éplucherai la liste de toutes les choses à effectuer pour faire passer cette application WCF classique de « hébergée en local » à « hébergée dans Windows Azure ». Une fois cela terminé, vous aurez le « backend » en place pour assurer une prise en charge multi-applications, multi-centres de données avec l'authentification intégrée.
- **Partie 3** : décrit l'assembly de la trousse d'outils personnalisé qui fournit « le ciment » pour connecter votre application WCF prenant en charge les revendications dans le cloud et votre ferme SharePoint. J'expliquerai comment utiliser cet assembly, je parlerai du contrôle personnalisé plutôt simple que vous devez créer (environ 5 lignes de code) et comment vous pouvez l'héberger dans une page du répertoire « _layouts » comme moyen de récupérer et d'effectuer le rendu des données dans un web part. L'intégralité du code source de l'exemple de contrôle personnalisé et de la page « _layouts » sera également publiée.
- **Partie 4** : le web part que j'inclus dans le CASI Kit. Il fournit une solution sans code, prête à l'emploi, qui vous permet de vous raccorder et de vous connecter à une requête asynchrone côté client pour récupérer des données à partir de votre service hébergé dans le cloud, et de les afficher ensuite dans le web part. Il dispose également « d'accroches » intégrées afin que vous puissiez le personnaliser un peu et utiliser vos propres fonctions JavaScript pour effectuer le rendu des données.
- **Partie 5** : un bref examen de quelques exemples d'applications qui illustrent quelques autres scénarios pour utiliser le contrôle personnalisé que vous avez construit dans **la troisième partie** dans d'autres cas de figures communs. L'un de ces scénarios utilisera le contrôle pour récupérer certains types de données utilisateur ou de données de configuration et les stocker dans le cache ASP.NET, puis les utiliser dans un web part personnalisé. L'autre scénario utilisera le contrôle personnalisé pour récupérer des données d'Azure et de les utiliser dans une tâche personnalisée ; dans le cas présent, un travail du minuteur SharePoint personnalisé. L'intégralité du code source de ces exemples d'applications sera également publiée.
- **Partie 6** : article complémentaire contenant quelques conseils sur l'utilisation de SQL Azure afin que vous puissiez l'intégrer facilement et rapidement dans vos projets de développement SharePoint 2010 avec Windows Azure.

Le CASI Kit : Partie 6 - Intégration de SQL Azure avec SharePoint 2010 et Windows Azure

Dans le CASI Kit, je vous ai fourni un certain nombre d'instructions et de lignes directrices, et d'outils pour vous permettre de vous connecter assez facilement et rapidement à Windows Azure depuis SharePoint 2010 tout en envoyant le jeton de l'utilisateur actuel, afin que vous puissiez prendre des décisions de sécurité affinées. L'application WCF d'origine que consommait le CASI Kit utilisait simplement une collection de données codée en dur qu'elle exposait. Dans une version ultérieure (et pas vraiment documentée dans les cinq premiers articles de la série sur le CASI Kit), j'ai mis à niveau la partie consacrée aux données de sorte à ce que le CASI Kit stocke et extrait des données dans l'espace de stockage des tables Windows Azure. À présent, je l'ai amélioré davantage en créant les données dans SQL Azure et en faisant en sorte que mon application WCF dans Windows Azure y extraie les données. À présent, on peut dire qu'il s'agit d'une suite d'applications « multi-cloud » : Windows Azure, SQL Azure

Copyright ® 2013 Steve Peschka. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

et (visiblement) SharePoint Online. Le but de cet article est juste de partager quelques conseils d'utilisation de SQL Azure pour que vous puissiez l'intégrer plus facilement dans vos projets de développement.

Conseils d'intégration

1 Vous devez impérativement procéder à une mise à niveau vers SQL Server 2008 R2 afin de pouvoir ouvrir les bases de données SQL Azure avec l'outil SQL Server Management Studio. Vous pouvez, certes, techniquement faire fonctionner SQL Server 2008, mais il y a de nombreuses et hasardeuses étapes de contournement à mettre en œuvre pour réaliser cela. Dans SQL Server 2008 R2, tout est déjà intégré et vous pourrez en tirer un meilleur parti. Si toutefois vous souhaitez passer par la voie de l'implémentation de la solution de contournement pour SQL Server 2008, consultez le lien suivant : **http://social.technet.microsoft.com/wiki/contents/articles/994.develop-and-deploy-with-windows-azure-sql-database.aspx**. Il y a plusieurs conseils judicieux dans cet article qui est donc à découvrir absolument en toutes circonstances.

2 Prévoyez d'utiliser T-SQL pour tout faire. Les outils graphiques ne sont pas disponibles pour fonctionner avec les bases de données, les tables, les procédures stockées SQL Azure, etc. Là encore, une chose que j'ai trouvée très utile, puisque je ne suis pas un vrai spécialiste de T-SQL, est de commencer par simplement créer une base de données SQL Server locale. Dans l'outil SQL Server Management Studio, je crée deux connexions - l'une vers mon instance SQL locale et l'autre vers mon instance SQL Azure. Je crée les tables, les procédures stockées, etc. dans l'instance SQL locale de manière à pouvoir utiliser les outils graphiques. Ensuite, j'utilise le script *[whatever Sql object] As…CREATE to…New Sql Query Window*. Cela génère le script SQL pour créer mes objets, et je peux ensuite le coller dans une fenêtre de requête que j'ai ouverte vers ma base de données SQL Azure.

3 Ce point est important - le délai d'attente par défaut n'est généralement pas suffisant pour se connecter à SQL Azure. Vous pouvez le modifier si vous utilisez les classes .NET SqlClient dans la chaîne de connexion, c'est-à-dire, ajoutez « Connection Timeout=30; ». Si vous utilisez SQL Server Management Studio, cliquez sur le bouton « Advanced » dans la boîte de dialogue dans laquelle vous saisissez le nom du serveur et vos informations d'identification, puis modifiez la valeur du paramètre à au moins 30. La valeur par défaut est de 15 secondes, ce qui est souvent insuffisant, mais une valeur de 30 secondes semble faire l'affaire. Malheureusement, il n'y a pas de méthode pour modifier le délai d'attente de connexion d'un type de contenu externe (c'est-à-dire, la définition d'une application BDC) vers une source de données SQL Azure.

4 N'utilisez pas le compte administrateur de la base de données pour établir la connexion à vos bases de données (c'est-à-dire, le compte qui vous est attribué pour créer la base de données elle-même). Créez un compte distinct pour travailler avec les données. Malheureusement, comme SQL Azure ne prend en charge que les comptes SQL, vous ne pouvez pas utiliser directement l'identité de l'utilisateur demandeur pour prendre des décisions sur l'accès aux données. Vous **pouvez** contourner cela en utilisant une application WCF qui met en avant plan les données dans SQL Azure et en utilisant l'authentification basée sur les revendications dans Windows Azure, ce qui correspond exactement au modèle qu'utilise le CASI Kit. En outre, cela demande quelques étapes supplémentaires pour créer un login qui peut être utilisé pour se connecter aux données d'une base de données spécifique. Voici un exemple :

```
--create the database first
CREATE DATABASE Customers

--now create the login, then create a user in the database from the login
CREATE LOGIN CustomerReader WITH PASSWORD = 'FooBarFoo'
CREATE USER CustomerReader FROM LOGIN CustomerReader

--now grant rights to the user
GRANT INSERT, UPDATE, SELECT ON dbo.StoreInformation TO CustomerReader

--grant EXECUTE rights to a stored proc
GRANT EXECUTE ON myStoredProc TO CustomerReader
```

5 Vous devez créer des règles de pare-feu dans SQL Azure pour chaque base de données dont vous disposez afin d'autoriser les communications provenant des différents clients. Si vous souhaitez autoriser la communication à partir d'autres services Azure, vous devez alors créer la règle de pare-feu MicrosoftServices (ce que SQL Azure propose de faire pour vous lorsque vous créez la base de données pour la première fois), qui est une plage de départ allant de 0.0.0.0 à 0.0.0.0. Si vous ne créez pas cette règle, aucune de vos applications Windows Azure ne pourra lire, ajouter ou éditer des données depuis vos bases de données SQL Azure ! Vous devez également créer une règle de pare-feu pour autoriser les communications avec les

Copyright ® 2013 Steve Peschka. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

serveurs de tout type que vous utilisez pour le routage vers l'Internet. Par exemple, si vous disposez d'un câble, d'un routeur DSL ou d'un serveur RRAS, vous voudrez alors utiliser votre adresse externe ou votre adresse NAT pour quelque chose telle que RRAS.

Ceux la devraient être de bons conseils pour rendre vos données opérationnelles.

Accès aux données

Accéder aux données elles-mêmes depuis votre code personnalisé - application WCF Windows Azure, web part, etc. - est heureusement plus ou moins similaire à l'extraction des données depuis un serveur SQL Server local. Voici un exemple de code rapide dont j'examinerai ensuite certaines parties plus en détail :

```
//set a longer timeout because 15 seconds is often not

//enough; SQL Azure docs recommend 30

private string conStr = "server=tcp:foodaddy.database.windows.net;Database=Customers;" +

"user ID=CustomerReader;Password=FooBarFoo;Trusted_Connection=False;Encrypt=True;" +

"Connection Timeout=30";



private string queryString = "select * from StoreInformation";

private DataSet ds = new DataSet();



using (SqlConnection cn = new SqlConnection(conStr))

{

SqlDataAdapter da = new SqlDataAdapter();

da.SelectCommand = new SqlCommand(queryString, cn);

da.Fill(ds);

//do something with the data

}
```

À ce stade, il n'y a pas grand-chose, à l'exception de la chaîne de connexion, qui mérite davantage d'explications. Les principaux éléments qu'il convient de noter, dont notamment les différences d'une connexion classique vers un serveur SQL Server local, sont :

- server : faites précéder le nom de la base de données SQL Azure de « tcp: ».
- Trusted_Connection : devrait avoir la valeur false car vous n'utilisez pas la sécurité intégrée.
- Encrypt : devrait avoir la valeur true pour toute connexion établie vers une base de données hébergée dans le cloud.
- Connection Timeout : comme décrit ci-dessus, le délai d'attente par défaut est de 15 secondes et cela n'est souvent pas suffisant ; à la place je le définis à 30 secondes ici.

Un autre scénario que je voulais mentionner brièvement ici est l'utilisation de la migration des données. Vous pouvez utiliser l'outil BCP fourni avec SQL Server pour déplacer des données d'un serveur SQL Server local vers SQL Azure. La routine de base pour migrer des données est comme ceci :

1. Créer un fichier de format : ceci est utilisé à la fois pour l'exportation des données locales et l'importation de celles-ci dans le cloud. Dans cet exemple, je suis en train de créer un fichier de format pour la table Region de la base de données Test, et de l'enregistrer dans le fichier *region.fmt*.

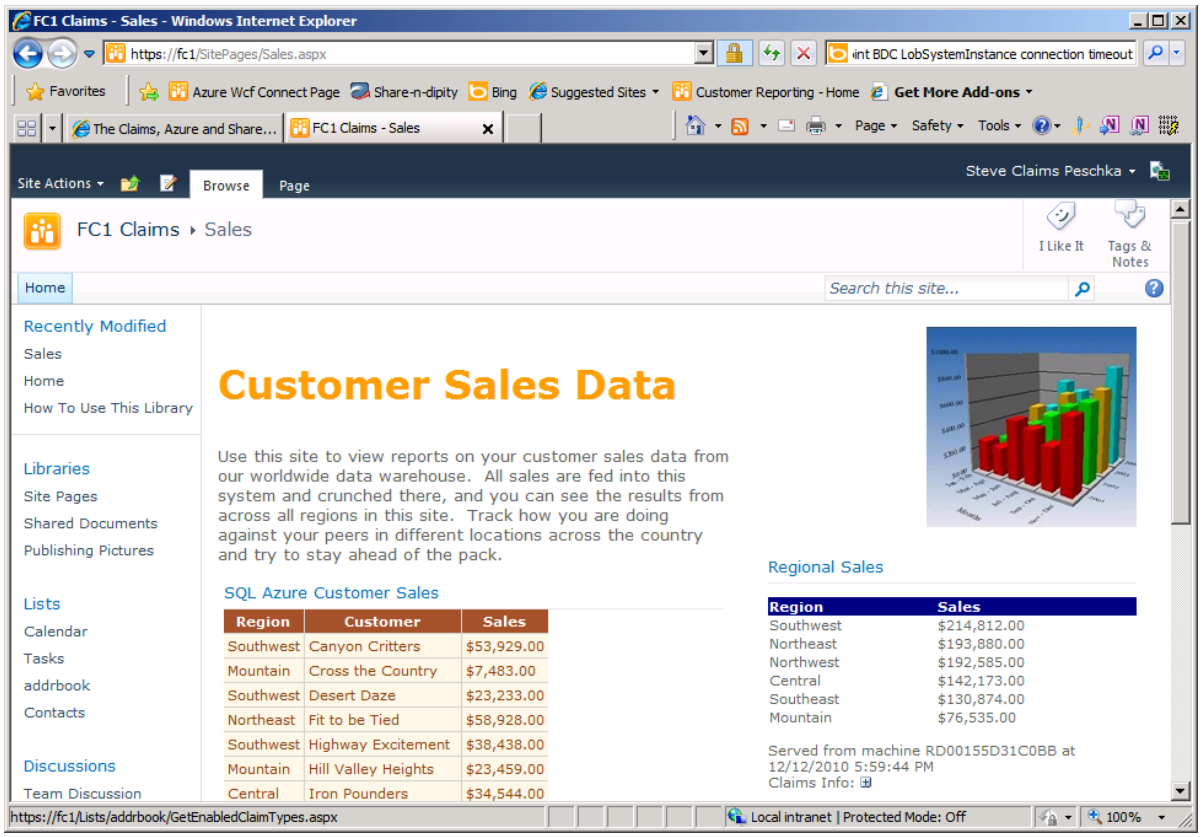
Copyright ® 2013 Steve Peschka. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

bcp Test.dbo.Region format nul -T -n -f region.fmt

2 Exporter des données depuis le serveur SQL Server local : dans cet exemple, je suis en train d'exporter les données de la table Region de la base de données Test, dans un fichier appelé *RegionData.dat*. J'utilise le fichier de format *region.fmt* que j'ai créé à l'étape 1.
bcp Test.dbo.Region OUT RegionData.dat -T -f region.fmt

3 Importer des données vers SQL Azure : la chose principale à noter ici est que lorsque vous êtes en train d'importer des données dans le cloud, vous **devez** inclure le « @servername » avec le paramètre du nom d'utilisateur ; l'importation échouera sans cela. Dans cet exemple, je suis en train d'importer les données dans la table Region de la base de données Customers de SQL Azure. Je suis en train d'importer du fichier de format RegionData.dat dans lequel j'ai exporté mes données locales. Mon nom de serveur (le paramètre -S) correspond au nom de la base de données SQL Azure. Pour mon nom d'utilisateur (le paramètre -U), j'utilise le format « username@servername » comme décrit plus haut. Je lui demande d'utiliser le fichier de format region.fmt que j'ai créé à l'étape 1 et j'ai défini une taille de lot (le paramètre -b) de 1 000 enregistrements à la fois.
bcp Customers.dbo.Region IN RegionData.dat -S foodaddy.database.windows.net -U speschka@foodaddy.database.windows.net -P FooBarFoo -f region.fmt -b 1000

C'est tout pour cet article ! Espérons que cela vous a permis de mieux comprendre les mécanismes de base de la création d'une base de données SQL Azure et de la connexion ainsi que l'utilisation de cette dernière dans votre site SharePoint. À signaler également que j'ai utilisé le CASI Kit pour extraire ces données via mon application « frontend » WCF Windows Azure vers SQL Azure et dont j'ai effectuées le rendu dans mon site SharePoint. J'ai moi-même suivi toutes les instructions figurant dans les cinq premiers articles de la série afin d'effectuer et de valider l'ensemble des opérations qui y sont décrites, et j'ai trouvé que tout était exact, à quelques exceptions près. Il y avait quelques corrections mineures que j'ai effectuées dans la **troisième partie** ainsi que l'ajout d'une section supplémentaire sur le dépannage. Mais, au total, il m'a fallu 30 minutes pour créer un nouveau contrôle personnalisé et peut être 15 minutes pour créer un composant Visual WebPart. J'ai utilisé le web part du CASI Kit pour afficher un ensemble de données SQL Azure de l'application WCF, et dans le composant Visual WebPart, j'ai créé une instance du contrôle personnalisé par programmation pour extraire un dataset que j'ai ensuite lié à une grille ASP.NET. J'ai tout rassemblé dans une page d'exemple qui me semble effectivement pas mal et qui pourrait être étendue pour afficher assez facilement les données dans plusieurs autres formats. Cette page ressemble à ceci :



Copyright ® 2013 Steve Peschka. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Conclusion

Ceci conclut donc cet article dans lequel nous avons vu quelques conseils sur l'utilisation de SQL Azure afin que vous puissiez l'intégrer facilement et rapidement dans vos projets de développement.

Liens

Vous trouverez en pièce jointe de cet article, un fichier compressé qui contient la version Word de l'article en anglais.

- **CASI Kit Partie 6.zip**

Le projet CASI Kit sur CodePlex : **ici**

Remerciements

Je tiens ici à remercier **Steve Peschka** de m'avoir autorisé à traduire son article.
Je remercie **xyz** pour sa relecture technique et ses propositions.
Je remercie également **abc** pour sa relecture orthographique et ses propositions.

Copyright ® 2013 Steve Peschka. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.