Numéro 3 | Novembre 2003

Generation PB

Bulletin d'information PowerBuilder

Bienvenue pour ce troisième numéro de Generation PB

Un numéro riche en nouvelles sur les produits et les événements par l'équipe PowerBuilder de Sybase

Que s'est-il passé depuis le deuxième numéro ? Un été long et chaud pour la plupart d'entre nous, au cours duquel s'est déroulé Techwave, la Conférence annuelle des utilisateurs Sybase, à Orlando en août.

Lors de Techwave, Pocket PowerBuilder 1.0, attendu depuis longtemps, a été lancé et la dernière roadmap de PB a dévoilé les prochaines nouveautés et fonctionnalités qui viendront enrichir PowerBuilder.

PB 9.0 étant la première pierre de l'UDE (environnement de développement unifié), les versions ultérieures consolideront l'engagement de Sybase à proposer un outil qui colle aux technologies émergentes telles que le développement à base de composants et le développement web, la modélisation et la gestion du cycle de vie (avec PowerAMC), les services Web, le Wi-Fi et l'intégration des applications.

Dans ce numéro, vous allez apprendre à accéder à différents composants à partir de votre application Pocket PowerBuilder (PPB) et découvrir DataWindow.NET, un nouveau produit PowerBuilder, dont le programme Beta sera bientôt disponible, ainsi que les dernières nouvelles sur nos produits et événements. Lisez la presse car je vais donner des interviews sur PowerBuilder dans la presse informatique européenne.

L'avenir sourit à PowerBuilder.

Steven Dunn - *Directeur des ventes PB, EMEA*

La tournée de lancement européenne de PB 9 a affiché complet

Cordiali saluti da Milano – séminaire de lancement de PB 9 organisé par Sybase Italie

Saludos de Barcelona – séminaire de lancement de PB9 d'AST

Grüße von Hamburg – séminaire PBUG de PowerPeople

Quel voyage étonnant : 12 conférences organisées en 2 mois avec plus de 900 participants au total.

Nous souhaitons profiter de l'occasion pour adresser tous nos remerciements aux organisateurs, aux intervenants et bien entendu aux utilisateurs de PowerBuilder pour avoir assuré la réussite de cet événement.

La question maintenant est : avez-vous mis en pratique ce que nos évangélistes techniques vous ont enseigné ?

Nous espérons que les démonstrations des solutions (services Web, DataWindow XML, PBNI, PocketPowerBuilder) vous ont été utiles dans vos projets.

Nous espérons également vous revoir prochainement. Consultez la page 6 pour connaître les événements prévus.

Dans le prochain numéro :

Vous en apprendrez encore plus sur les fonctions de PowerBuilder, de Pocket PowerBuilder et de DataWindow .NET. Pour vous abonner, envoyez un e-mail à l'adresse softwarecentre@sybase.com

Qui est qui? Cao Lin

Permettez-nous de vous présenter Cao Lin, Directeur de l'ingénierie

Cao Lin
Directeur de l'enginierie

Cao Lin est titulaire d'un MEng et d'un Ph. D. Son expérience en développement logiciel couvre les domaines du système de base de données bancaire, du système d'aide à la décision, du générateur de code C++, du système d'extraction de données etc... Basé à Singapour, il travaille au développement et à la gestion de PowerBuilder depuis plus de 5 ans. Il a commencé par le développement de PowerBuilder 6.5 pour les versions chinoise et coréenne chez Sybase. Il a constitué les excellentes équipes QA et de développement Asie de PowerBuilder en 2000. En collaboration avec l'équipe PowerBuilder Amérique du Nord, il a piloté le lancement réussi de PowerBuilder 8.0. En tant que directeur de l'ingénierie, il est en charge du développement mondial de PowerBuilder. Cao Lin et son équipe ont fait de PowerBuilder 9.0 la version la plus passionnante depuis PB 5.0. Voir que PowerBuilder 9.0 contribue à redonner confiance aux clients l'encourage dans son travail.

Pour lui, PowerBuilder est l'outil de développement d'applications de la prochaine génération : il synthétise et simplifie les technologies complexes utilisées par les développeurs actuels et à venir pour qu'ils soient plus productifs et compétitifs en matière de développement d'applications sur des plates-formes ouvertes. En particulier, les fonctions clés suivantes devraient enrichir la version 10 de PowerBuilder ou les versions ultérieures. L'approche conception-développement itérative permettra aux développeurs de modéliser et de rétro-concevoir le code pour avoir une image complète du diagramme de classes. DataWindow.NET permettra aux développeurs d'utiliser le DataWindow avec d'autres langages .NET. Le compilateur .NET permettra aux développeurs PB de tirer parti d'autres langages .NET dans l'IDE PowerBuilder. Le DataWindow pour Java permettra aux développeurs Java d'utiliser le DataWindow sur une plate-forme Java. Plus tard, les développeurs PB disposeront des options nécessaires pour générer des applications PB, .NET ou Java dans l'IDE PowerBuilder en vue du déploiement. Les développeurs pourront également décider de déployer leurs applications sur des ordinateurs de bureau ou des dispositifs sans fil.

Conseils & astuces

10 conseils pour migrer les applications vers PowerBuilder 9.0

par Lee Cheetham, Consultant Services Professionnels de Sybase

1) Utiliser l'Assistant de Migration pour identifier les fonctions et les événements obsolètes. Cet assistant analyse les bibliothèques de PowerBuilder afin de détecter les éventuels problèmes de syntaxe avant la migration de l'application. La fonction Obsolete Syntax est disponible depuis la version 6. L'Assistant de Migration se trouve dans l'onglet « Tool » de la boîte de dialogue de l'objet « New ».

2) Fichiers PSR, une application échouera si elle essaie d'ouvrir un fichier PSR créé dans une version antérieure de PowerBuilder (antérieure à la version 8.0 build 7063 ou à la version 7.0 build 10102). Cet échec est lié à une modification de la fonction SaveAsASCII. Pour résoudre ce problème, il suffit de recréer le fichier PSR dans un build ultérieur de PowerBuilder.

3) De nouveaux mots réservés (Try, Catch, Finally, Throw, Throws) ont été introduits dans PowerBuilder 8.0 dans le cadre de la gestion des exceptions. L' Assistant de Migration va identifier toutes les occurrences de ces mots réservés. Pour remédier à ce problème, remplacez toutes les instances de « Try, Catch, Finally, Throw, Throws » dans votre application d'autres termes en utilisant une version antérieure de PowerBuilder.

4) Le support pour la programmation distribuée n'est plus disponible depuis PowerBuilder 8.0. L'objet « Transport » est devenu obsolète. Les événements « Connectionbegin » et « Connectionend » de l'application ne sont plus déclenchés. La solution consiste à déployer vos NVO côté serveur sur EAServer.

5) Le traitement des événements SystemError a été modifié depuis PowerBuilder 7. Dans cette version, lorsqu'une erreur se produit, l'événement « SystemError » est déclenché et, une fois qu'il a été traité, vous revenez dans le script où l'erreur s'est produite. Dans PowerBuilder 8 et 9, lorsqu'une erreur se produit, la pile d'appels est déroulée, puis l'événement « SystemError » est déclenché, mais une fois qu'il a été traité, le traitement ne reprend **PAS** au point où cette erreur s'est produite. Repérez et gérez les erreurs en local en utilisant la nouvelle fonction de gestion des exceptions disponible dans les versions 8 & 9.

6) La fonction IsValid a été modifiée dans PowerBuilder 8 afin de retourner « FALSE » si l'objet n'est pas instancié ou si un objet non valide a été passé. Avant la version 8, si un objet non valide était passé, l'événement « SystemError » était déclenché.

7) Windows n'hérite plus de l'icône Application, il vous faut désormais coder ou paramétrer la propriété sur « AppIcon! ».

8) TreeView Event, l'événement **« rbuttonup! »** n'est plus déclenché, cela a été introduit dans PowerBuilder 7.

9) ListView, l'événement **« pbm_rbuttonup! »** n'est plus déclenché lorsque vous cliquez avec le bouton droit sur l'élément ListView, mais il se déclenchera si vous cliquez sur la zone blanche dans la ListView. Pour remédier à cela, utilisez le nouvel événement **« pbm_contextmenu! »** qui se déclenche toujours lorsque le bouton droit de la souris est utilisé.

10) Les cibles web (web targets) qui utilisent la commande WebDataWindow doivent être modifiées pour utiliser le nouveau composant HTMLGenerator90. Cliquez avec le bouton droit sur la commande Web DataWindow Design Time sur votre page web et sélectionnez les propriétés. Dans l'onglet « HTML Generator », remplacez la référence HTMLGenerator80 par HTMLGenerator90. Une fois que HTMLGenerator a été modifié, les propriétés de la connexion à la base de données sont parfois perdues, il est donc important de vérifier l'onglet « Connection » et de vous assurer que le paramétrage de la connexion à la base de données est toujours correct après le changement.

DataWindow .NET – La puissance du DataWindow au service de Visual Studio .NET

Par David Avera, ingénieur d'étude, Sybase, Inc.

Les programmeurs PowerBuilder chevronnés savent parfaitement quelles sont la puissance et la capacité de la commande DataWindow et du DataStore. Ce logiciel breveté (brevets américains 5566330, 5752018, 5832481, 5937415) est décrit comme une « invention qui comporte un système de définition d'une interface de base de données permettant à un programmeur d'applications de définir graphiquement, d'afficher et d'utiliser [un objet visible] le data window pour manipuler indirectement les données dans une base de données applicative. » Ensuite, il est indiqué que « l'invention comporte un objet programme [un objet DataWindow] fournissant une interface entre un gestionnaire de base de données afin de gérer une table de base de données et un programme applicatif client externe à l'objet interface. »

La consultation en ligne des demandes de brevets est idéale pour remédier à l'insomnie. La chose importante à retenir est que le DataWindow est le plus beau joyau de PowerBuilder, l'un de ses éléments majeurs qui le différencie de tous ses concurrents. A ce jour, personne d'autre n'a réussi à concevoir un produit qui rivalise avec le DataWindow, tant en capacité qu'en convivialité.

Pourquoi DataWindow .NET ?

Le DataWindow est une exclusivité PowerBuilder depuis le début des années 90, à l'exception des commandes DataWindow ActiveX et DataWindow pour Java™. La technologie .NET de Microsoft a donné à Sybase l'occasion de l'étendre aux développeurs utilisant des langages et des outils Microsoft. Il s'agit d'une communauté très large, comprenant (essentiellement) des développeurs Visual Basic mais également de plus en plus de développeurs en C#.

En outre, de plus en plus de développeurs PowerBuilder sont mandatés pour créer certaines de leurs applications avec des outils .NET. DataWindow .NET leur permettra de maintenir leurs investissements dans la technologie DataWindow, sans avoir à apprendre et à perfectionner l'utilisation des DataGrid, DataView, DataSet ainsi que des différents types d'adaptateurs et de connecteurs de Microsoft.

Jusqu'à présent, les utilisateurs d'outils de développement Microsoft utilisaient un mélange de composants tels que les Datagrid, les DataView, les DataAdapters pour chaque base de données prise en charge, les DataSets, les objets Sql Connection et Sql Command. Il existe un marché tiers actif qui étend ces composants, les rendant plus faciles à utiliser (et plus semblables au DataWindow).

Sybase proposera une meilleure solution pour l'accès aux données, le reporting et la manipulation des données : DataWindow .NET.

Le produit DataWindow .NET offrira un ensemble de services complets des composants familiers DataWindow et DataStore de PowerBuilder dans l'IDE Visual Studio. Les utilisateurs seront en mesure de programmer DataWindow .NET avec n'importe quel langage .NET.

De quoi est composé DataWindow .NET ?

DataWindow .NET sera composé d'un jeu de DLL qui forment un serveur front-end et un serveur back-end DataWindow. DataWindow Builder sera une des fonctionnalités intégrées afin de créer et de maintenir les définitions des temps de conception du DataWindow.

Le serveur front-end DataWindow .NET fournit l'interface entre l'application Visual Basic ou C# et le serveur DataWindow. Il établit une correspondance entre les méthodes, événements et propriétés classiques de DataWindow et l'application. Le front-end garantit également que les applications .NET n'interagiront directement qu'avec du code .NET pur. Le serveur back-end DataWindow est invisible aux applications, la seule contrainte étant que les DLL se trouvent sur le chemin de chargement.

Le serveur DataWindow gère le chargement et la présentation des objets DataWindow et gère la communication avec les bases de données. Les fonctions traditionnelles du DataWindow, telles que la gestion des mémoires tampons, le tri, le filtrage, l'exportation de données vers une variété de formats de fichiers et d'expressions DataWindow, sont toutes exécutées dans ce serveur.

Comment utiliser DataWindow .NET ?

Entrons dans le vif du sujet !

En partant de l'hypothèse que vous disposez de certaines définitions DataWindow dans un fichier PBL ou PBD, nous allons créer une application C# simple en utilisant l'assistant Visual Studio New Project. Sélectionnez « Visual C# Projects » et « Windows Application ». Nous allons utiliser un DataWindow dans l'application, nous connecter à la base de données et extraire et afficher les données. Nous allons également ajouter un gestionnaire d'événements pour l'événement DataWindowRetrieveEnd.

Note de l'auteur : les images et le code suivants sont tirés de la version pré-alpha et ne sont pas forcément identiques à ceux du produit final.

Figure 1. Affichage initial du masque de saisie avec la Toolbox

Comme vous pouvez le voir dans la Toolbox, il existe un composant DataWindowControl. Cliquez sur cet élément et déposez-le dans le masque de saisie. Vous obtiendrez un rectangle vide identifiant l'emplacement du DataWindowControl.

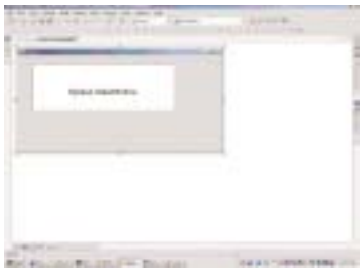


Figure 2. Masque de saisie avec un DataWindowControl vide

Ajoutez quelques propriétés au DataWindowControl

Figure 3 *Onglet des propriétés du DataWindowControl.*

Cliquez sur le DataWindow pour en faire la commande active et affichez l'onglet Properties.
Vous y trouverez certaines propriétés familières des objets DataWindow. Choisissez « LibraryList » pour définir l'emplacement à partir duquel la définition du DataWindow doit être chargée. Ensuite, sélectionnez la propriété DataObject pour sélectionner un objet DataWindow spécifique.

Sélectionnez un fichier PBL (ou PBD) contenant des définitions de DataWindow

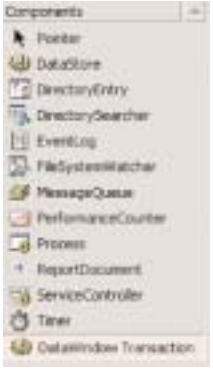
Figure 4. *Boîte de dialogue de la LibraryList.*

Sélectionnez un DataWindow dans le fichier PBL ou PBD

Figure 5. *Fenêtre de sélection de DataWindowObject*

Le DataWindowControl du masque de saisie ressemble maintenant à ceci :

Figure 6. *Le DataWindowControl avec des en-têtes*



Ajoutez un DataWindow Transaction à l'application.

Figure 7. *Boîte à outils Visual Studio*

Dans l'onglet Components de la Toolbox, sélectionnez le composant DataWindow Transaction, déposez le dans le masque et définissez ses propriétés.

Figure 8. *Propriétés de la transaction*

(suite des Fig. 7 et 8) En faisant glisser et en déposant la transaction, et en définissant ses propriétés, vous instanciez automatiquement un objet transactionnel (que nous appellerons SQLCA) dans l'application. Bien entendu, cela peut également être effectué dans le code C#. Si vous examinez le code généré par l'IDE, vous verrez :

```
this.DWC1 = new
Sybase.DataWindow.DataWindowControl();
this.SQLCA = new
Sybase.DataWindow.Transaction(this.components);
```

et

```
this.DWC1.DataWindowObject = "d_dept";
this.DWC1.LibraryList = "E:\\PB Diamond
Workspaces\\externa.pbl";
```

Maintenant pour du codage réel

Dans le constructeur de masque de saisie, vous pouvez définir la connexion à la base de données :

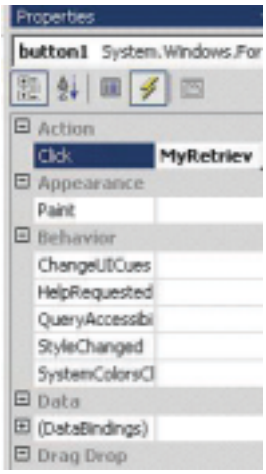
```
public Form1()
{
      //
      // Required for Windows Form Designer
support
      //
      InitializeComponent();
//
// TODO: Add any constructor code after
InitializeComponent call
      //
      SQLCA.Connect( );
}
```

Maintenant, passons à la génération du code pour l'extraction du DataWindow. Sélectionnez une commande Button dans l'onglet Windows Form de la Toolbox et déposez-la sur le masque. Ajoutez des propriétés, telles que « Retrieve DataWindow » pour le texte du bouton, sélectionnez les événements dans l'onglet des propriétés, et tapez le nom de la méthode que vous souhaitez utiliser.

Figure 9. *Masque de saisie avec un bouton.*

Figure 10. *Onglet des propriétés du bouton montrant l'événement cliqué.*

L'IDE affichez la fenêtre du code du masque de saisie, avec le code de texte prédéfini suivant :



Suite. Fig.10

```
private void
MyRetrieve(object sender,
System.EventArgs e)
{
}
```

Insérez le code suivant dans la méthode MyRetrieve :

```
DWC1. SetTransObject
(SQLCA):
DWC1. Retrieve ( );
```

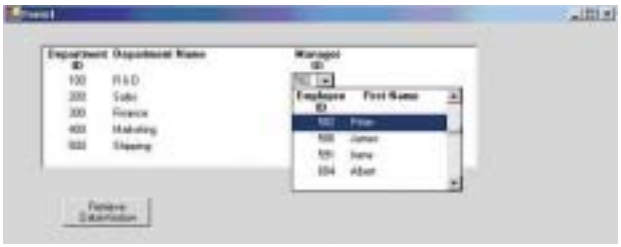
```
intDataWindowControl.Retrieve (params
object[] Argslist)
Retrieves rows from the database for the
DataWindow control. If arguments are included,
the argument values are used for the retrieval
arguments in the SQL SELECT statement for the
DataWindow object.
```

Cela est un exemple de la capacité de la technologie Intellisense de Visual Basic d'afficher de la documentation pour une méthode en passant la souris sur le verbe.

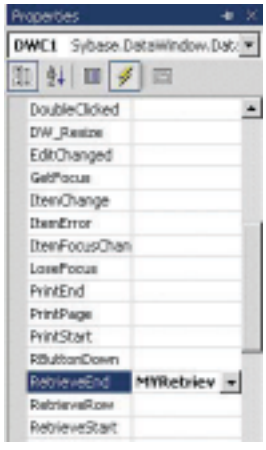
Enfin, lancez l'application et cliquez sur le bouton « Retrieve DataWindow » :

Figure 11. *Masque de saisie affiché à l'exécution.*

La table Department familière a été extraite et affichée. Veuillez noter que la fonction Drop down DataWindows.



Supposons que nous souhaitons savoir quand se termine l'extraction et connaître le nombre de lignes extraites



Retournez dans l'IDE, sélectionnez DataWindowControl afin d'en faire la commande active et affichez l'onglet des propriétés.

Figure 12. *Onglet des propriétés de DataWindowControl montrant les événements.*

Vous verrez s'afficher du code de texte prédéfini dans la fenêtre du code :

```
private int MYRetrieveEnd(object sender,
Sybase.DataWindow.DataWindowEvents.RetrieveEndArg
s args)
{
      return 0;
}
```

Ajoutez le code suivant à la méthode :

```
String Rows = args.RowsRetrieved.ToString( );
MessageBox.Show ("Rows Retrieved = "+ Rows,
"Retrieve End");
```

Faites une extraction avec une méthode de traitement d'événements RetrieveEnd

Figure 13. *Masque de saisie s'exécutant avec une boîte de message Retrieve End*

Cet exemple montre combien il est facile d'utiliser un DataWindow existant dans une application C# .NET. Il est tout aussi facile de le faire dans Visual Basic .NET.

Au-delà de l'exemple simple ci-dessus, nous allons fournir la véritable puissance du DataWindow aux programmeurs .NET. Un de nos spécialistes PowerBuilder en interne est en train de porter les exemples du DataWindow PowerBuilder dans Visual Basic avec succès pour un produit en phase pré-alpha. Ces exemples montreront qu'une utilisation plus complexe du DataWindow sera toute aussi efficace qu'une utilisation simple.

Quels éléments seront pris en charge dans le produit final ?

La quasi-totalité des méthodes, propriétés et événements habituels des DataWindows seront identiques à ce qu'ils sont dans PowerBuilder, avec quelques exceptions :

1. Les événements DataWindow seront codés en utilisant le modèle d'événement .NET, les mêmes arguments seront disponibles, mais spécifiquement pour les objets des arguments des événements.

2. La notation Dot n'est pas prise en charge dans DataWindow .NET. Les méthodes SetProperty et GetProperty seront disponibles.

3. Le style Richtext DataWindow n'est pas pris en charge.

Synthèse

Dans cet article, j'ai essayé de vous montrer combien il sera facile d'utiliser DataWindow .NET dans une application Visual Studio en utilisant C#. Si vous connaissez Visual Basic, le processus est tout aussi simple. Dans l'exemple, nous avons montré comment cliquer sur et déposer les éléments à partir de la Toobox, spécifier des valeurs de propriétés pour ces éléments, ajouter un peu de code, et nous avons créé une application qui extrait et affiche des données de la base de données. Au-delà de cet exemple simple se cache la véritable puissance du DataWindow, qui est :

● Formatage des données ● saisie des données masquées ● SaveAs n'importe quel fichier pris en charge notamment en XML ● puissantes options de reporting telles que les groupes et les bandes de synthèse ● une variété de formats de reporting, de gabarits, free form, de tabulations croisées, de graphiques, etc. ● mise à jour de la base de données, y compris des procédures stockées ● expressions DataWindow ● tri – filtrage ● et bien plus encore.

Avec DataWindow .NET, Sybase offre aux développeurs Visual Studio une alternative aux objets DataGrid, DataView et aux autres objets Microsoft. Sybase offre, par ailleurs, aux programmeurs PowerBuilder appelés à utiliser la technologie .NET dans une application un moyen de mettre à profit leur expérience du DataWindow dans un nouvel environnement.

Dernières nouvelles !

Les événements à venir : à vos calendriers !

18 novembre 2003 – **Norvège : séminaire Sybase.** Pour plus d'informations et vous inscrire, rendez-vous sur le site **www.sybase.no**

3 décembre 2003 **GB : conférence STUN.** Pour plus d'informations et vous inscrire, rendez-vous sur le site **www.stun.org.uk**

4 décembre 2003 – **Danemark : rencontre PBUG.** Pour plus d'informations et vous inscrire, rendez-vous sur le site **www.ravenholm.dk**

9 décembre 2003 – **Belgique : Sybase Techserie, Kinepolis Bruxelles.** Pour plus d'informations et vous inscrire, rendez-vous sur le site **www.sybase.be**

10 décembre 2003 – **Luxembourg: Séminaire PB 9, Sofitel Luxembourg.** Pour plus d'informations et vous inscrire, rendez-vous sur le site www.sybase.be

Nouveaux ouvrages sur PB 9 :

Deux nouveaux ouvrages sur PowerBuilder 9.0, rédigés par des spécialistes, viennent de sortir et portent sur le **développement client/serveur avancé et le développement distribué.**

PowerBuilder 9 Advanced Client/Server Development par Bruce Armstrong, Millard F. Brown III

PowerBuilder 9 Internet and Distributed Application Development par William Green, John D. Olson

Egalement disponible, **Mobile and Wireless Design Essentials** par Martyn Mallick, Consultant en solutions iAnywhere.

Mobile and Wireless Design Essentials est le premier et seul ouvrage qui offre une analyse détaillée de toutes les technologies mobiles et sans fil actuelles en adoptant le point de vue d'un développeur logiciel.

Notification de fin de support de PowerBuilder 7.0.x

Nous vous encourageons à passer à PowerBuilder 8.0 ou 9.0 (édition Enterprise Professional ou Desktop) dans la mesure où la version 7.0.x (Enterprise, Professional et Desktop) n'est plus supportée.

Infos Version actuelle / EBF

DESCRIPTION	VERSION	DATE	TYPE
PB Enterprise - 9.0.1 (6533) Maintenance release	9.0.1	05/09/2003	Maint/Mise à jour
PB Professional - 9.0.1 (6533) Maintenance release	9.0.1	05/09/2003	Maint/Mise à jour
PB Desktop - 9.0.1 (6533) Maintenance release	9.0.1	05/09/2003	Maint/Mise à jour
Fichiers Runtime localisés PB Enterprise - 9.0.1 (6533)	9.0.1	08/10/2003	Maint/Mise à jour
Fichiers PCF localisés PB Enterprise - 9.0.1 (6533)	9.0.1	08/10/2003	Maint/Mise à jour
PB Enterprise - 8.0.4 (10501) Maintenance release	8.0.4	06/10/2003	Maint/Mise à jour
PB Professional - 8.0.4 (10501) Maintenance release	8.0.4	08/10/2003	Maint/Mise à jour
PB Desktop - 8.0.4 (10501) Maintenance release	8.0.4	08/10/2003	Maint/Mise à jou
Version EBF PB Enterprise - 7.0.3 (10312)	7.0.3	20/08/2003	EBF/Patch

Remarque : le build EBF de PB v7.x est le dernier livrable pour cette version qui a été arrêtée le 15/7/2003:

Remarque : vous trouverez davantage d'informations sur www.downloads.sybase.com

Pocket PowerBuilder, SOAP & PocketSOAP

Par Ian Thain, Evangéliste technique, IPG, Sybase Inc.

Des composants très divers sont disséminés au sein de votre entreprise et vous devez y accéder à partir de votre application Pocket Powerbuilder ? Je ne compte pas discourir sur SOAP & les services Web dans la mesure où il existe une littérature très abondante sur ces thèmes que vous pouvez consulter, notamment si vous souffrez d'insomnies. Je vais me contenter de présenter ces deux technologies et l'utilisation qui nous pouvons en faire.

SOAP

Le protocole SOAP (Simple Object Access Protocol) est souvent considéré comme l'épine dorsale d'une nouvelle génération d'applications distribuées indépendantes des plateformes et langages: les services Web. Comme son nom le suggère, c'est un protocole léger, basé sur XML, pour échanger des informations dans un environnement distribué. Le protocole SOAP comprend trois parties : une **enveloppe** qui définit un cadre pour décrire le contenu du message et comment le traiter ; des **règles de codage** pour les types de données ; et une **convention** pour les appels et les réponses de procédure à distance (RPC).

Services Web

Les interfaces et les liaisons publiques d'un service Web sont définies et décrites en XML et sont identifiées par un URL. La définition et la description sont définies en WSDL (Web Services Definition Language) et peuvent être découvertes par d'autres systèmes informatiques via UDDI. Une fois découverte, l'interaction se produit grâce aux messages XML transmis par des protocoles Internet (SOAP) d'une manière définie dans sa définition.

PocketSOAP

PocketSOAP est un composant COM (Component Object Model) client SOAP open source, conçu à l'origine pour les PocketPC, et qui est maintenant développé pour PocketPC et Windows 95, 98, Me, NT 4.0 et 2000. PocketSOAP peut passer des appels RPC à des serveurs SOAP mis en œuvre avec 4s4c, ROPE, Apache SOAP, SOAP::Lite, DM's SOAP/Perl et le XMethods SOAP Server. Il est doté d'un client HTTP (Hypertext Transfer Protocol) pour effectuer des requêtes SOAP basées sur HTTP, mais d'autres transports peuvent être utilisés. PocketSOAP est distribué sous la licence MPL (Mozila Public License). PocketSOAP est doté des fonctions suivantes : support du codage de la section 5 de SOAP, support pour les services SOAP de style document/literal (tels que ASP.NET), support des attachements via à la fois DIME et SOAP with Attachments, support pour SOAP, support HTTP 1.1 dont les connexions persistantes, SSL, les proxies, l'authentification, l'authentification des proxies, les redirections, les cookies et la compression. D'autres fonctionnalités sont à découvrir sur **http://www.pocketsoap.com**

Pocket PowerBuilder & PocketSOAP

Pocket PowerBuilder et PocketSOAP interagissent via l'interface de PPB pour PocketSOAP. Il s'agit d'un wrapper DLL externe (dans la mesure où PPB n'a pas d'objet COM, c-à-d OleObject), qui fournit un accès simple à PocketSOAP, via son interface COM. Le modèle de l'API est que l'objet service est d'abord créé, et cela retourne le descripteur d'accès. Ensuite, vous pouvez définir les divers attributs dans n'importe quel ordre souhaité. En interne, les attributs sont simplement stockés pour une utilisation ultérieure. Ensuite, vous procédez à l'appel SOAP. Cette opération utilise les attributs précédemment définis. Enfin, vous détruisez l'objet service, ce qui supprime tous les objets COM précédemment créés. Tout cela est généré en utilisant le système SOAP. Aussi, pour faire cela avec PocketSOAP, récupérez les logiciels sur http://www.pockesoap.com et installez-les à la fois sur le PC et sur le Pocket PC.

EAServer

Prenons un exemple d'utilisation de PocketSOAP dans Pocket PowerBuilder pour appeler un service Web dans EAServer 4.2. Ce service Web est en fait un EJB qui extrait les prix d'anciens stocks dans une base de données ASA via un cache de connexion.

Cet EJB est exposé comme un service Web en le définissant avec la Web Services Toolkit (WST) dans EAServer. Cela nous permet d'exposer tout composant de EAServer comme un service Web. Le WST générera l'ensemble du WSDL nécessaire.

Voici le code nécessaire pour accéder au service Web via PocketSOAP.

```
string sMethod = "getPFShareList"
string sEndPoint  =
"http://localhost:10000/WEBSERVICES/SOAP"
string sNameSpace = "PFShareListAll/PFShareList"
int    cRetBufLen = 128
string sArgs
string sResult
long   lHandle
int    iRet

// create & set attributes on for the soap call
iRet = PocketSoap_Create( true, REF lHandle )
wf_message( iRet, "PocketSoap_Create")

iRet = PocketSoap_SetEndPoint( lHandle, sEndPoint )
wf_message( iRet, "PocketSoap_SetEndPoint")

// preallocate the result string
sResult = Space( cRetBufLen )

// make a "simple call"
iRet =  PocketSoap_SimpleCall( lHandle, sNameSpace,
sMethod, "", "" )
wf_message( iRet, "PocketSoap_SimpleCall")

// get the Result
iRet =  PocketSoap_GetResult( lHandle, cRetBufLen, REF
sResult )
wf_message( iRet, "PocketSoap_GetResult")

// clean up
iRet =  PocketSoap_Destroy( lHandle )
wf_message( iRet, "PocketSoap_Destroy")

wf_message (integer ai_ret, string as_method)
If ai_ret <> 0 Then
MessageBox("Return code", String(ai_ret) + " " +
as_method)
End if
```

Les 3 éléments importants à définir sont le **EndPoint**, le **Namespace** et la **Method**. Le EndPoint indique un emplacement spécifique pour accéder au service en utilisant un protocole et un format de données spécifiques. Le Namespace indique le service Web spécifique et, à l'évidence, la méthode est celle que nous souhaitons appeler dans ce service Web. Dans l'exemple, nous allons utiliser la méthode Pocketsoap_SimpleCall qui exécute un appel synchrone sur l'URL du EndPoint avec un seul argument. Cet argument est une paire nom/valeur et il est facultatif. Pour un appel plus complexe, il existe une autre méthode appelée PocketSoap_Call, qui utilise des arguments sous forme de paires nom/valeur mais représentées comme une seule chaîne.

```
Name1 ~t value1 ~t data-type1 ~r~n
Name2 ~t value2 ~t data-type2 ~r~n
Name3 ~t value1 ~t data-type1 ~r~n
Etc
```

PocketSOAP se chargera de générer le message SOAP et d'analyser la réponse. L'appel SOAP et la réponse du serveur peuvent être visualisés en utilisant l'outil SOAP Util d'Apache (http://www.apache.org). Vous lancez cette opération en utilisant la ligne de commande suivante :

```
java org.apache.soap.util.net.TcpTunnelGui
10000 ithain-home 8080
```

Cela permet à l'outil d'intercepter toutes les requêtes qui sont effectuées au localhost:10000, de les afficher puis de les rediriger vers ithain-home:8080, de les afficher de nouveau et de les rediriger vers l'appelant.

Pour EAServer 4.x, il est nécessaire de paramétrer l'attribut **SoapAction**. En effet, dans EAServer 4.x nous utilisons cet attribut, depuis notre implémentation propriétaire des services Web, même si cela est compatible avec une autre implémentation telle qu'Apache. Cela étant dit, d'autres implémentations requièrent également SoapAction. Mais, il n'est pas nécessaire de vous en préoccuper dans la mesure où dans EAServer 5.0 l'implémentation des services Web sera Axis et il ne sera pas nécessaire de paramétrer SoapAction.

XMethods

Voici un exemple d'appel d'un service Web sur XMethods (http://xmethods.net). Il utilise un service Web qui est un devis décalé de 20 minutes, mis en oeuvre via GLUE. Dans XMethods, il existe un certain nombre d'autres services Web mis en oeuvre en utilisant MS .NET, Delphi, SOAPLite, WASP Server for Java, ColdFusion, BEA WebLogic, ApacheSOAP, AXIS et bien d'autres.

```
string sMethod = "getPFShareList"
string sEndPoint  =
"http://localhost:10000/WEBSERVICES/SOAP"
string sNameSpace = "PFShareListAll/PFShareList"
int    cRetBufLen = 128
string sArgs
string sResult
long   lHandle
int    iRet

// create & set attributes on for the soap call
iRet = PocketSoap_Create( true, REF lHandle )
wf_message( iRet, "PocketSoap_Create")

iRet = PocketSoap_SetEndPoint( lHandle,
sEndPoint )
wf_message( iRet, "PocketSoap_SetEndPoint")

// preallocate the result string
sResult = Space( cRetBufLen )

// make a "simple call"
iRet =  PocketSoap_SimpleCall( lHandle,
sNameSpace, sMethod, "", "" )
wf_message( iRet, "PocketSoap_SimpleCall")

// get the Result
iRet =  PocketSoap_GetResult( lHandle,
cRetBufLen, REF sResult )
wf_message( iRet, "PocketSoap_GetResult")

// clean up
iRet =  PocketSoap_Destroy( lHandle )
wf_message( iRet, "PocketSoap_Destroy")

wf_message (integer ai_ret, string as_method)
If ai_ret <> 0 Then
MessageBox("Return code", String(ai_ret) + " "
+ as_method)
End if
```

Cette fois, l'outil SOAP Util a été lancé en utilisant la ligne de commande suivante :

```
java org.apache.soap.util.net.TcpTunnelGui
10000 services.xmethods.net 9090
```

En outre, le EndPoint a été défini comme la chaîne sEndPoint = http://localhost:10000/soap plutôt que http://services.xmethods.net/soap dans le fragment de code ci-dessus. Vous pouvez également constater, à

partir de la requête, que cette mise en oeuvre ne requiert pas de définir un attribut SoapAction.

Conclusion

Au cours de ces dernières années, nous vous avons demandé de partitionner vos applications, de réutiliser le code et de tirer parti de votre serveur d'applications préféré (EAServer). PocketSOAP peut être utilisé pour accéder aux composants exposés n'importe où comme des services Web et il est accessible à partir des applications Pocket PowerBuilder via une interface PPB.

« Cet article a été initialement publié dans PowerBuilder Developer's Journal (Vol. 10, numéro 8) ». Pour vous abonner à PBDJ, inscrivez-vous maintenant à l'adresse subscribe@sys-con.com ou rendez-vous sur le site www.sys-con.com pour plus d'informations.