

Tp WinDev Numéro 7

	<p>Objectifs : Création d'un Webservice</p> <p>Paramétrage d'un serveur Web, Création du Service Web, Création du client consommateur, Approche XML, SOAP ...</p>
---	---

<p>Outils : Easyphp ou Apache seul. Ce TP a été crée avec WinDev 9, l'adaptation à la version 8 ou 7 est mineure, c'est plus souvent un problème de localisation des menus.</p>
--

Le but de cet exercice est de vous faire pénétrer dans le monde merveilleux et surtout à la mode des Web-Services. En effet l'heure actuelle est à répartition des charges et des serveurs, c'est dans ce contexte qu'interviennent les Web-Services. Mais d'abord quelques définitions :

1. Les Webservices :

Dans ce chapitre, nous allons vous présenter les WebServices : c'est-à-dire pourquoi les WebServices ont été créés et à quelle demande répond cette nouvelle technologie. Nous verrons ensuite le fonctionnement d'un Webservice

1.1.Présentation :

Auparavant pour mettre en place des applications distribuées, il fallait utiliser des technologies assez complexes telles que COM. Certes ces technologies étaient abordables pour un développeur, mais il fallait que le développeur passe du temps à établir un protocole de transmission. Les WebServices sont alors apparus pour faciliter tout d'abord la tâche des développeurs. Avant toute chose, Microsoft, contrairement aux idées reçues, n'a pas créé les WebServices mais Microsoft a participé avec de grandes entreprises telles que IBM, SUN ... à la standardisation des WebServices. Ceci montre bien que la technologie des WebServices est une technologie très jeune, ce qui bien sûr peut être un inconvénient pour son intégration au sein des entreprises. Mais les plus grands spécialistes prévoient une « explosion » de l'utilisation des WebServices toutes technologies confondues (.NET, Java ...).

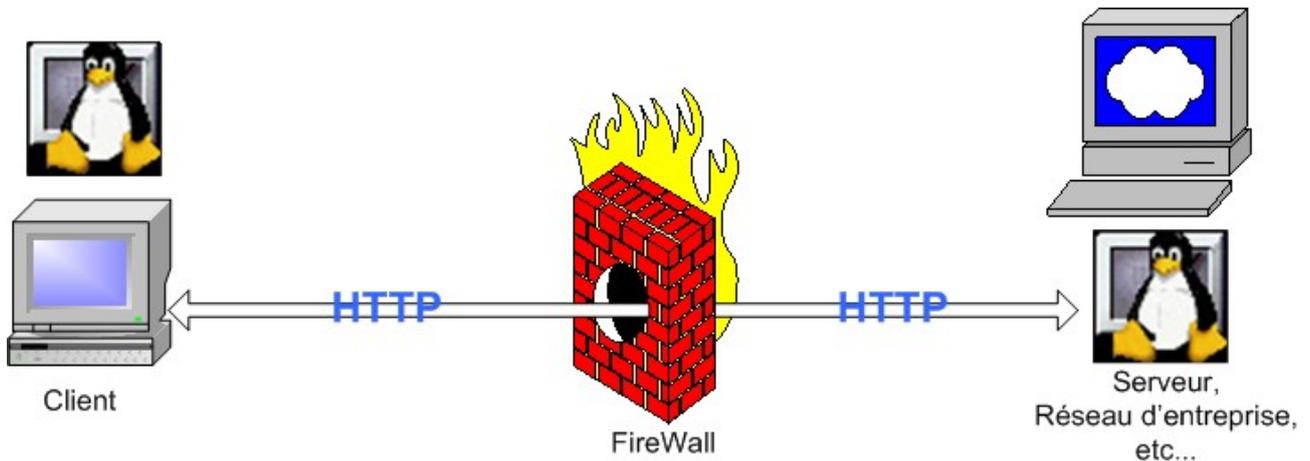
1.2. Fonctionnement des WebServices

L'un des plus gros avantages des WebServices est qu'ils reposent sur des protocoles standardisés. Cela permet que cette technologie soit exploitable par de nombreux langages. En effet, les WebServices se reposent sur des protocoles tels que XML et http, donc SOAP. Pour vulgariser ce dernier protocole, SOAP permet de faire circuler du XML via du HTTP. Donc lorsqu'on interroge un Webservice, les données sont transmises en XML via le port 80 (HTTP). Rien de plus simple ensuite pour le développeur de traiter l'information reçue. A l'heure actuelle, la quasi totalité des langages informatiques supporte ces protocoles : ils disposent en effet de fonctions pour lire un fichier XML (Parseur XML). Donc un Webservice peut être utilisé via le langage Perl, PHP, Python, **Delphi, Cobol ...**

1.3 Pourquoi les WebServices ?

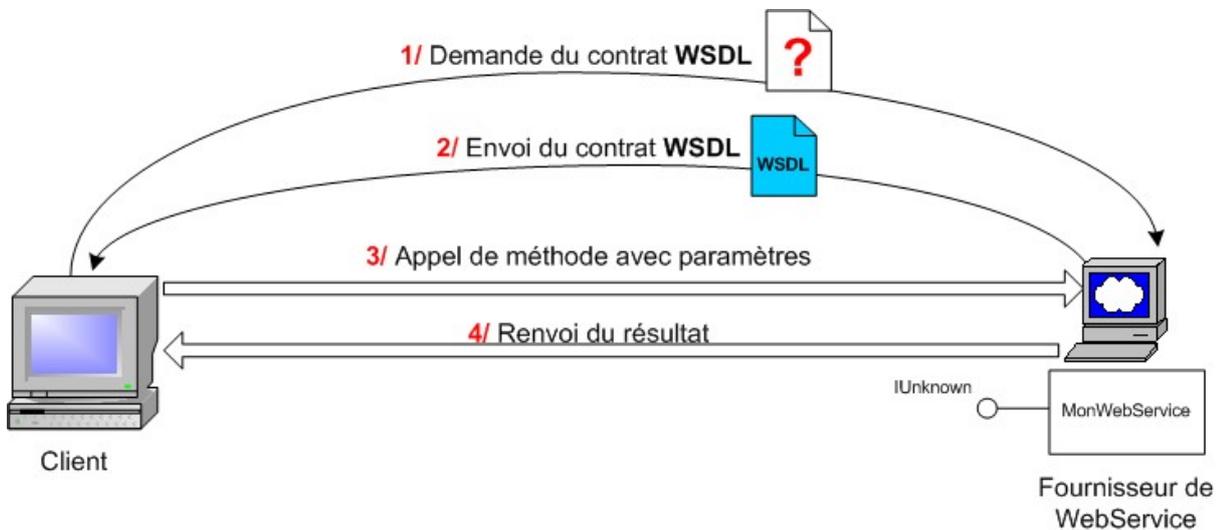
Comment faire communiquer des programmes tournant sur des machines distantes, des OS différents, développés par des compagnies différentes ? Comment dans ce cas de figure faire du remoting ? A moins d'utiliser des nombreux ponts qui existent, c'est quasi impossible. Bien évidemment, des ébauches de solutions ont été apportées et ont fait leur preuve (CORBA, COM, DCOM, ...)

Les WebServices sont eux universels et de plus le HTTP passe sans peine par un firewall ...



L'utilisation d'un Webservice peut se diviser en différentes étapes :

1. On demande au Webservice son contrat WSDL (Web Service Description Language) : c'est un document formalisé (XML, W3C) qui spécifie quels sont les méthodes pouvant être appelées sur ce Webservice.
2. Il le retourne, et on mémorise comment il marche (méthodes, format des appels, paramètres, valeurs retournées, etc...). Pour cela, on crée ce que l'on appelle en Java un classe stubs (souche). En C# et sous .NET, le terme consacré est classe proxy : généré par l'outil wsdl.exe, c'est une classe qui présente les mêmes méthodes que le Webservice, et qui permet de les appeler de manière synchrone ou asynchrone.
3. A l'utilisation : on appelle la méthode (SOAP+XML) désirée conformément au format précédemment acquis. Et ceci, tout simplement en instanciant et en utilisant ses méthodes. C'est transparent.
4. On récupère le résultat de la méthode, ou une erreur. Je vous souhaite le résultat, mais une erreur sera levée si le schéma WSDL du Webservice a changé !



Les étapes de l'utilisation d'un Webservice

Les WebServices passant (sauf si vous en décidez autrement) par http, peuvent utiliser d'autres protocoles que SOAP pour transporter des données. Mais ceci implique des restrictions, énoncées ci-dessous :

Quoi	POST	GET	SOAP
Transporter des types primitifs (Integer, Long, String, ...)	✓	✓	✓
Transporter des énumération	✓	✓	✓
Transporter des tableaux	✓	✓	✓
Objets			✓
Structures			✓
DataSets, fichiers XML, tableau de n'importe quoi			✓
Passage par référence			✓

NB : GET et POST sont les méthodes de passage de valeur propres à HTTP.

2. Soap

2.1 Présentation

Nous allons décrire dans ce chapitre le protocole SOAP et ses concurrents (COM, CORBA ...). En effet, comme nous l'avons vu dans le chapitre précédent, la technologie des WebServices repose en autres sur le protocole SOAP.

SOAP est un protocole adopté par le Consortium W3C. Le Consortium W3C crée des standards pour le Web : son but est donc de créer des standards pour favoriser l'échange d'information. Un standard veut tout simplement dire qu'il peut être accessible à tout le monde, et donc qu'il n'est pas propriétaire. Ce qui a pour conséquence qu'un protocole standard contrairement à un protocole propriétaire pourra être utilisé sous n'importe quelle plateforme.

Les spécifications du protocole SOAP sont disponibles à l'adresse suivante :

<http://www.w3.org/TR/SOAP/>

SOAP veut dire : Simple Object Access Protocol. Si l'on voulait traduire cette définition en français cela donnerait Protocole Simple d'Accès aux Objets. En effet, le protocole SOAP consiste à faire circuler du XML via du http sur le port 80. Cela facilite grandement les communications, car le XML est un langage standard et le port utilisé est le port 80, qui ne pose donc pas de problèmes pour les firewalls de l'entreprise, contrairement à d'autres protocoles. Tout comme la technologie des WebServices, le protocole SOAP est très jeune. Le protocole SOAP a été créé en septembre 98, avec la version 0.9, par trois grandes entreprises : Microsoft, UserLand et DevelopMentor. IBM n'a participé au protocole SOAP qu'à partir de la version 1.1 en avril 2000. C'est cette même année que SOAP a été soumis au W3C.

Depuis septembre 2000, SOAP 1.1 est en refonte complète pour donner jour à la version 1.2 avec un groupe de travail de plus de 40 entreprises ! Parmi ces 40 entreprises, on retrouve bien sûr Microsoft, IBM mais aussi HP, Sun, Intel ...)

2.2 Les autres protocoles... :

Jusqu'à la création du protocole SOAP, trois grands protocoles étaient utilisés : **COM et DCOM** :

Les protocoles COM (Component Object Model) et DCOM (Distributed Component Object Model) ont été écrits par Microsoft et permettaient de faciliter la communication entre les composants Windows. Il y a eu un portage de COM sous Unix, mais ce protocole n'a été utilisé que par des plateformes Windows et pour l'Intranet.

Les protocoles COM et DCOM n'étaient utilisés la plupart du temps que pour l'Intranet, car le port d'écoute des communications était statique : c'est-à-dire qu'on ne pouvait pas changer ce port et cela posait de gros problèmes de sécurité pour les entreprises qui voulaient utiliser ce protocole pour communiquer entre elles.

2.2.1. CORBA :

CORBA (Common Object Request Broker Architecture) a été créé par l'OMG (Object Management Group) pour faciliter la communication sous n'importe quelle plateforme. Ceci a été réalisé via un langage neutre de définition d'interface appelé IDL (Interface Definition Language) et un protocole commun de transport des données. Malheureusement, les spécifications de ce protocole sont très denses et l'architecture est donc au final très lourde à déployer.

2.2.2 RMI :

RMI (Remote Method Invocation) est un protocole très simple a utiliser et très efficace mais limité à l'environnement Java

3. La mise en œuvre

La présentation étant faite passons à l'action !

Pour pouvoir tester le Webservice sur votre machine de développement il va vous falloir un serveur Web en fonctionnement. Je vous propose de travailler avec Apache. Le plus simple est d'installer Easyphp. Une fois celui-ci installé sur votre poste vous devrez le configurer pour qu'il reconnaisse les Webservices WinDev. Suivez les étapes suivantes (tirées de l'aide en ligne de WinDev rubrique SOAP)

1. Ouvrez le fichier "httpd.conf" dans le bloc-notes Windows. Ce fichier est présent dans le sous-répertoire conf de votre installation Apache.
2. Recherchez la section concernant le support des objets partagés. Pour cela, recherchez :
soit la ligne suivante : " # Dynamic Shared Object (DSO) Support "
soit le mot-clé " LoadModule ".
Ajoutez la ligne :
 LoadModule windev_module <Répertoire d'installation de WinDev 9>\Donnees\WD90sapa.dll
(NB : Les utilisateurs de Windev8/7 modifieront en conséquence)
4. Recherchez la section concernant les " handlers " de requêtes. Pour cela, recherchez :
soit la ligne " # AddHandler allows you to map certain file extensions to "handlers",
soit le mot-clé " AddHandler ".
Ajoutez la ligne suivante :
 AddHandler windev-module .soap
5. Recherchez la ligne
"WHENEVER YOU CHANGE THE LOADMODULE SECTION ABOVE, UPDATE THIS TOO!"
et saisissez le caractère # devant le mot ClearModuleList :

Ces deux dernières opérations ne pourront ce faire qu'à la fin de l'exercice. Pensez y avant de crier que le webservice ne fonctionne pas !

6. Recherchez dans le fichier "httpd.conf" la section concernant le répertoire par défaut des fichiers. Pour cela, recherchez le mot clé " documentroot ".
Copiez dans le répertoire indiqué après le mot clé " documentroot " les fichiers suivants :

1. le fichier XML contenant la description du service Web XML.
2. la bibliothèque WinDev de votre application serveur ou du service WEB XML (fichier *.WDL).
3. l'application WDSOAPConfig.EXE présente dans le répertoire "Données" du répertoire d'installation de WinDev 9.
4. les DLL WinDev nécessaires à l'exécution de votre application. Vérifiez que les DLL suivantes sont sélectionnées : WD90IMG.DLL, WD90OBJ.DLL, WD90STD.DLL, et WD90VM.DLL.
5. les DLL WinDev spécifiques au serveur HTTP utilisé : WD90SAPA.DLL pour Apache

NB (Pour WinDev 8 recherchez les fichiers WD80IMG.DLL, WD80OBJ.DLL.....)

7. Lancez l'administrateur SOAP (application WDSOAPConfig.EXE). Cet administrateur est présent dans le répertoire d'installation de votre application (ou dans un répertoire spécifique s'il est déjà installé).

Paramétrez les différents éléments :

Temps avant de télécharger une WDL inutilisée :

si plusieurs bibliothèques (fichier WDL) correspondant à des applications SOAP serveur (ou à des services Web XML) sont présentes sur le poste, cette option permet de configurer le temps maximal d'attente avant de supprimer de la mémoire une bibliothèque inutilisée. Lors de cette suppression le code de fin de projet sera exécuté.

Nombre maximum de WDL en mémoire :

si plusieurs bibliothèques (fichier WDL) correspondant à des applications SOAP serveur (ou à des services Web XML) sont présentes sur le poste, cette option permet de configurer le nombre de WDL chargées simultanément en mémoire. Lorsque ce nombre est atteint, la bibliothèque la plus ancienne est automatiquement fermée. Lors de cette fermeture le code de fin de projet sera exécuté.

Enregistrer un journal (.LOG) :

Cette option permet d'enregistrer dans un fichier texte toutes les opérations réalisées sur le serveur SOAP. Pour chaque opération, la date et l'heure sont indiquées. Ce fichier peut par exemple contenir les messages suivants :

Chargement de la WDL X

Réception d'une requête : appel à la fonction X de la WDL Y

L'appel à la fonction X a échoué

L'appel à la fonction X a réussi

On atteint la limite de WDL en mémoire

Déchargement de la WDL X

Fichier journal :

Répertoire du serveur dans lequel le fichier journal doit être créé.

Localisation des WDL :

Répertoire du serveur où sont présentes les bibliothèques des applications Serveur SOAP.

Remarque : si le répertoire n'existe pas, les WDL seront recherchées dans le répertoire C:\modulessoap.

Localisation des DLL WinDev :

Répertoire du serveur où sont présentes les DLL WinDev utilisées par les applications Serveur SOAP.

Remarque : si le répertoire n'existe pas, les DLL seront recherchées dans le répertoire C:\modulessoap.

Localisation des fichiers en exécution :

Répertoire des fichiers Hyper File.

Lors du chargement de la bibliothèque du serveur SOAP :

- l'analyse associée au serveur SOAP sera automatiquement chargée.

- le répertoire des données spécifié dans l'analyse correspondra automatiquement à ce répertoire.

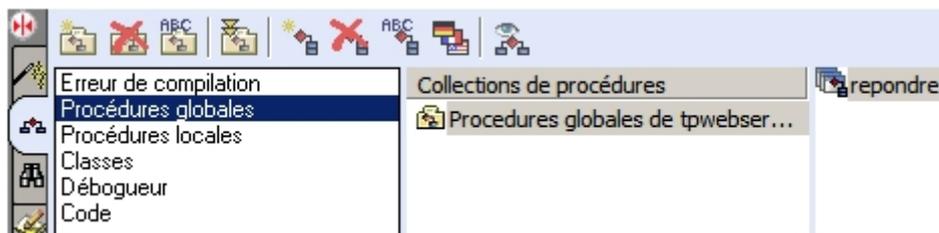
De l'action

Bon, on y va ?

Ok, c'est parti !

Créez un nouveau projet : TpWebService. Ce projet ne contiendra aucune analyse, ni aucunes fenêtres. Un Webservice étant comparable à une collection de procédures les fenêtres ne sont d'aucunes utilités.

Comme je viens de vous le dire un WebService est un ensemble de procédures ou de classes. Dans cet exercice nous allons utiliser les procédures globales. Créez en une nommée « Répondre »



Le code est le suivant :

```
RENGOYER "coucou"
```

Comme vous le voyez : on se fatigue ! En fait pour ce premier exercice on se contente d'illustrer les principes donc je vous prie de m'excuser pour la faiblesse de l'illustration !!! Vous ferez mieux plus tard ;-))

Vous venez de le comprendre que lorsqu'un client consommateur appellera la méthode « Répondre » du WebService il aura une chaîne de caractère « Coucou » comme réponse. C'est puissant, non ?!!!

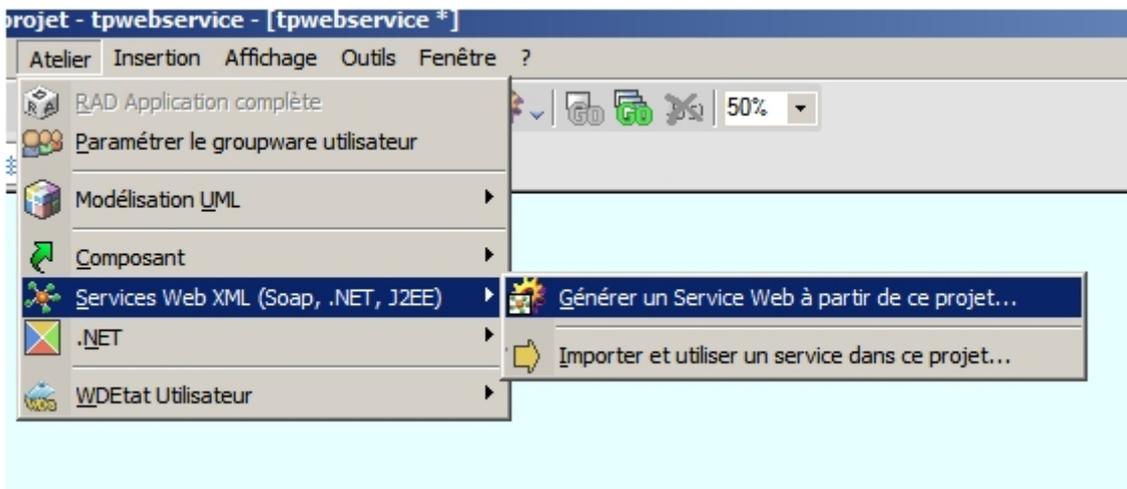
Bon, si je vous disais que le Webservice est fini, vous le croiriez ? et pourtant c'est vrai !. Ah non, il reste à le déployer, on va le faire de suite :

Pour Windev 7,5 :

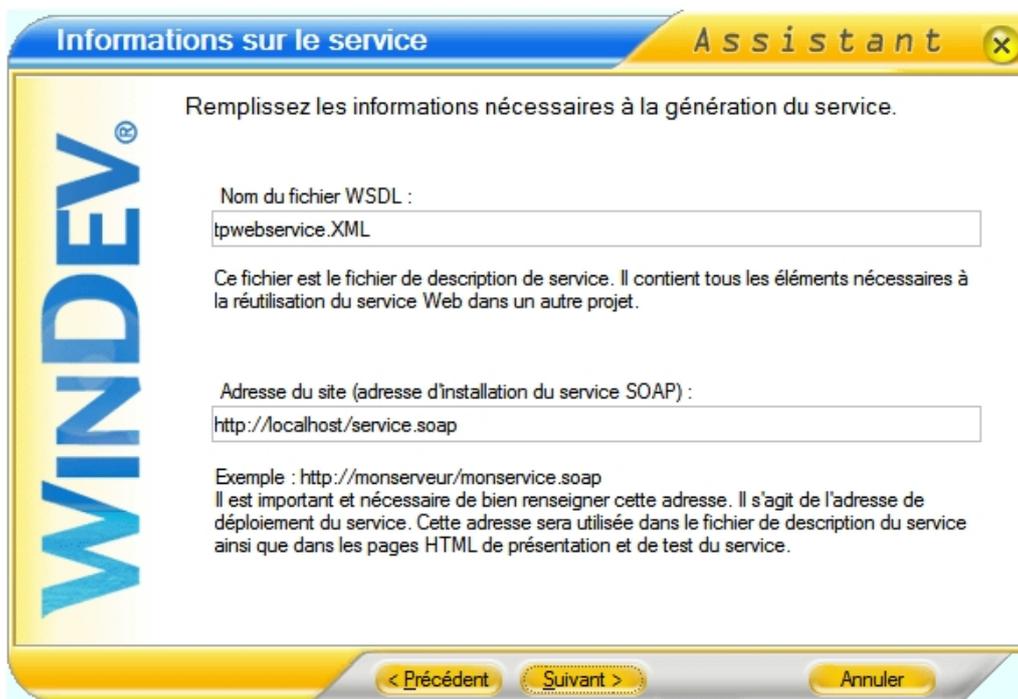
Débrouillez-vous, non mais !

Pour Windev 9 :

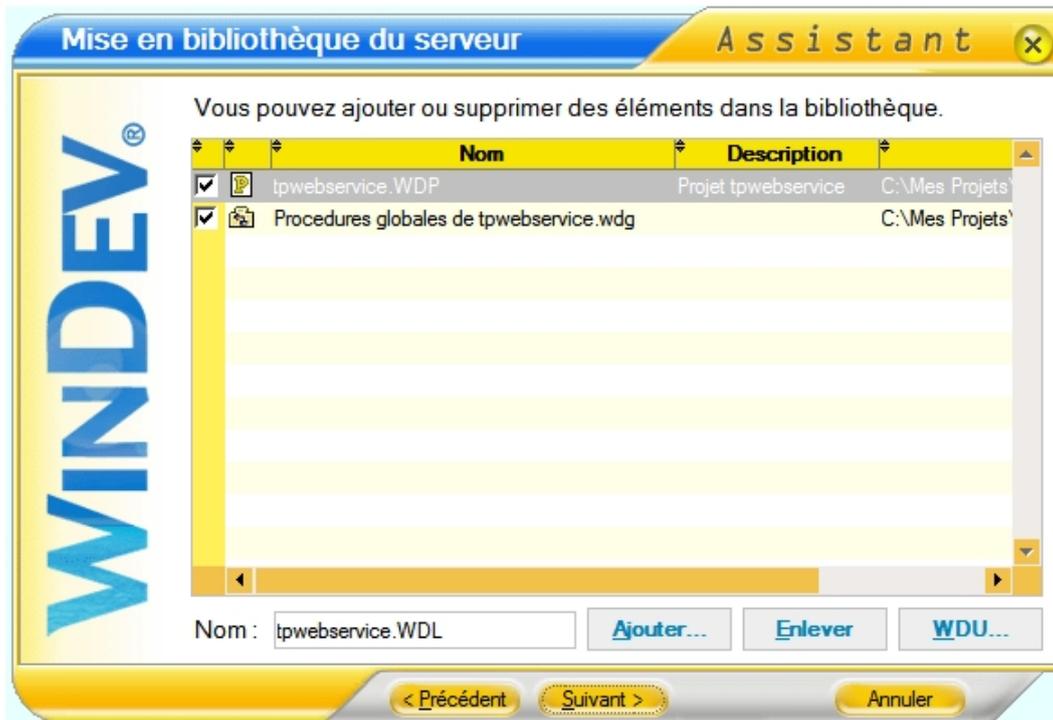
Allez dans le menu **Atelier** puis **Service Web XML (SOAP, .net, J2EE)** puis **Générer un service Web à partir de ce projet...**



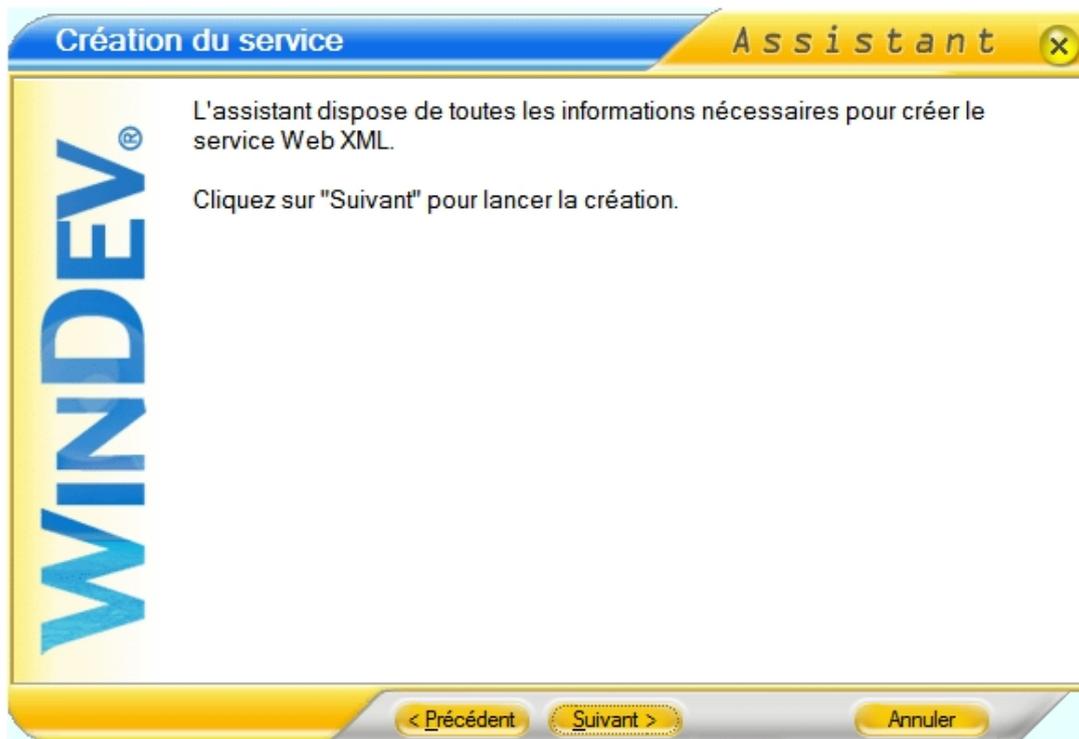
L'écran suivant lance l'assistant :



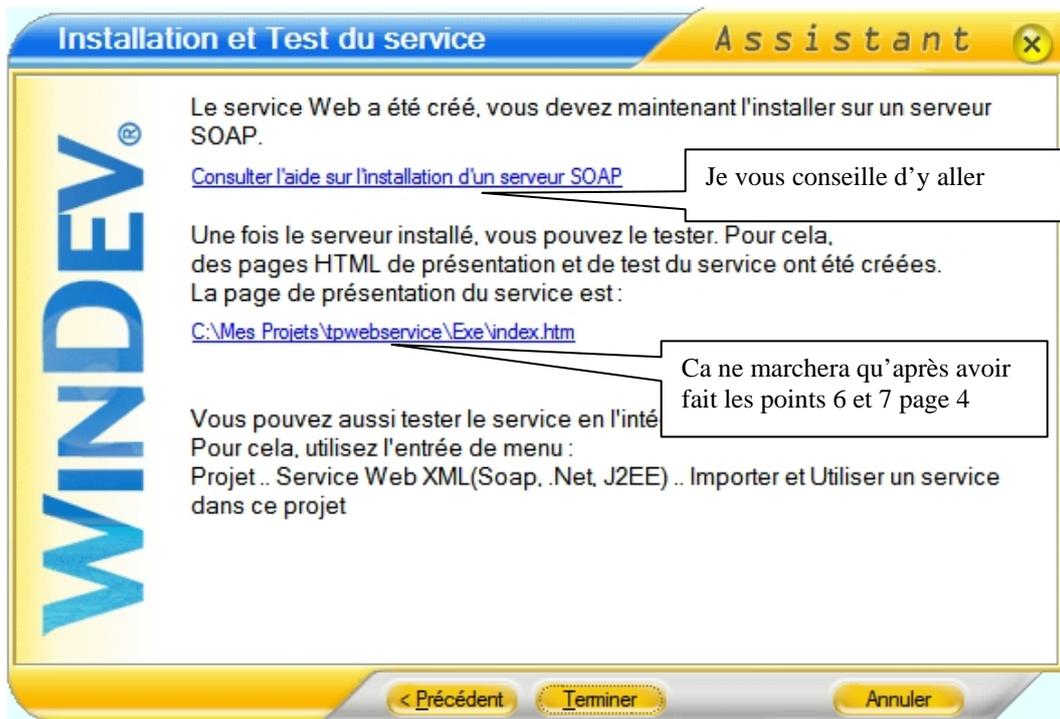
Remplissez les 2 champs et cliquez sur suivant :



Cliquez sur suivant.

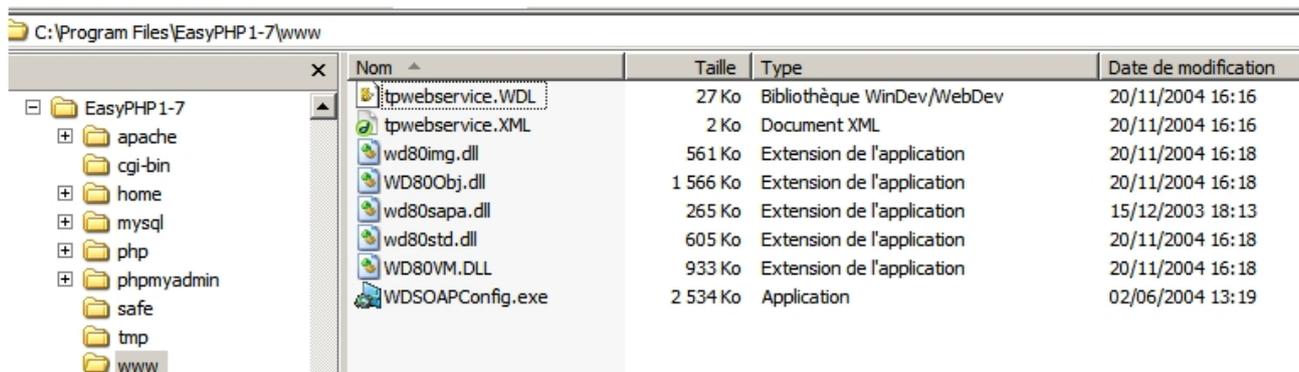


Ca deviendrait presque lassant ! Cliquez encore sur suivant.



Et voilà ! C'est fini. Le webservice est terminé. Mais pas son installation finale. Revenez aux points 6 et 7 Page 4.

Une fois les fichiers copiés dans le bon dossier apache, vous devriez obtenir ceci :



Lors de votre configuration dans le fichier «Httpd.Conf » apache, vous avez indiqué à Apache comment se comporter pour exécuter un Webservice SOAP Windev (Cf : loadmodule).

Nous retrouvons donc les librairies (.dll) spécifiques WinDev pour exécuter dans un contexte client/serveur les méthodes contenues dans tpwebservice.wdl.

Le fichier tpwebservice.xml est un fichier structuré contenant la description du Webservice . Editez le avec un éditeur de texte et vous allez en comprendre le but.

Fermez le projet du Webservice

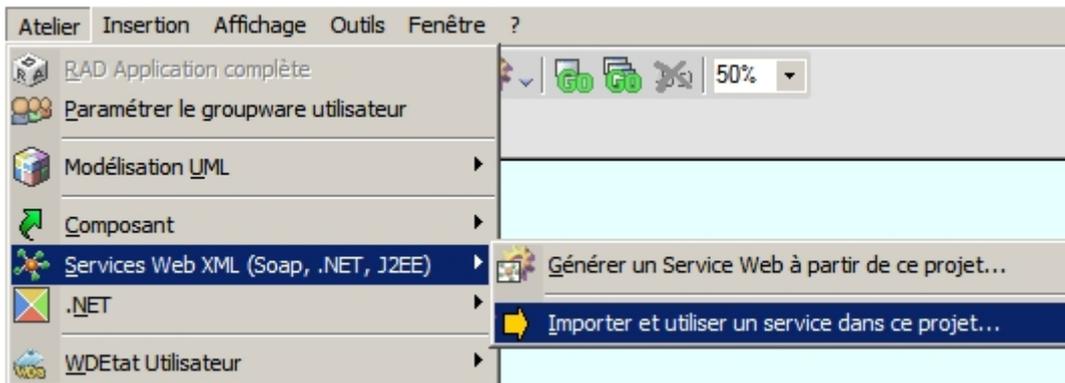
Maintenant, le morceau de bravoure : le client ou consommateur du Webservice.

Nous allons créer le client.

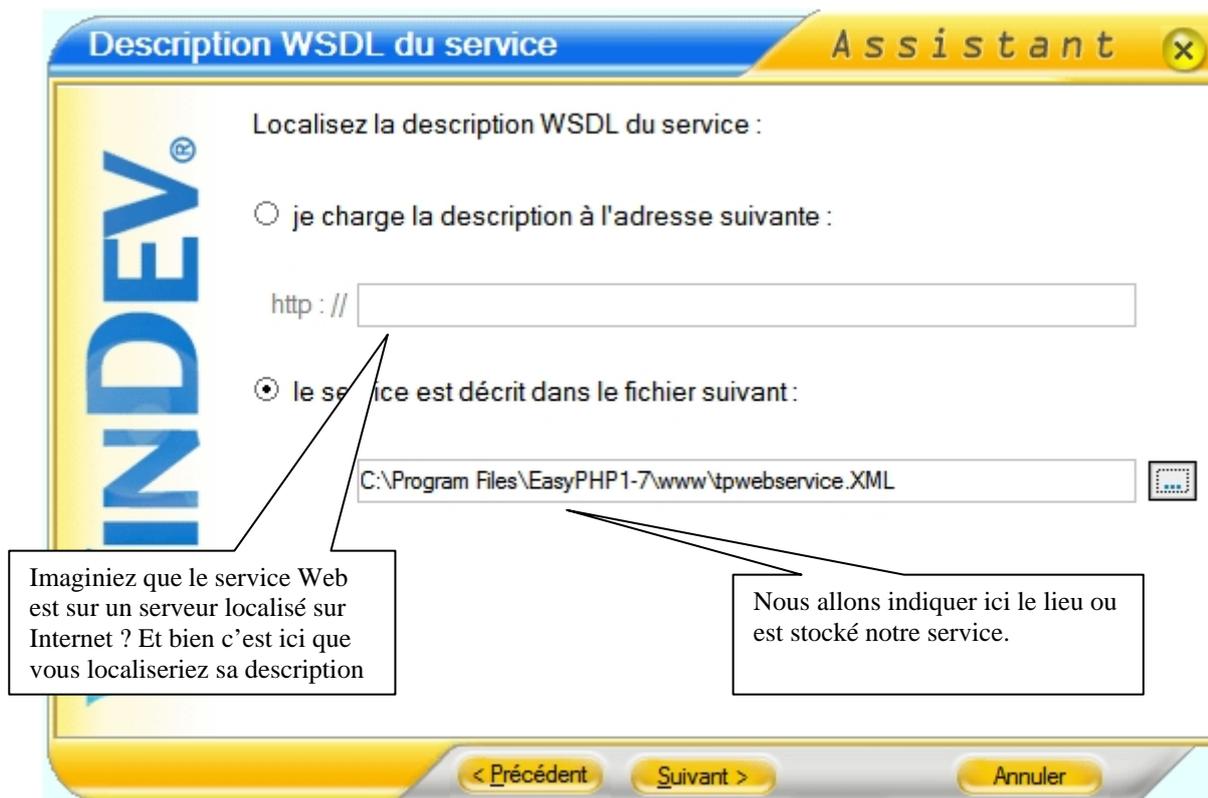
Fichier ->Nouveau ->Projet.

Vous nommerez ce projet TpclientSoap. Il n'utilisera pas d'analyse.

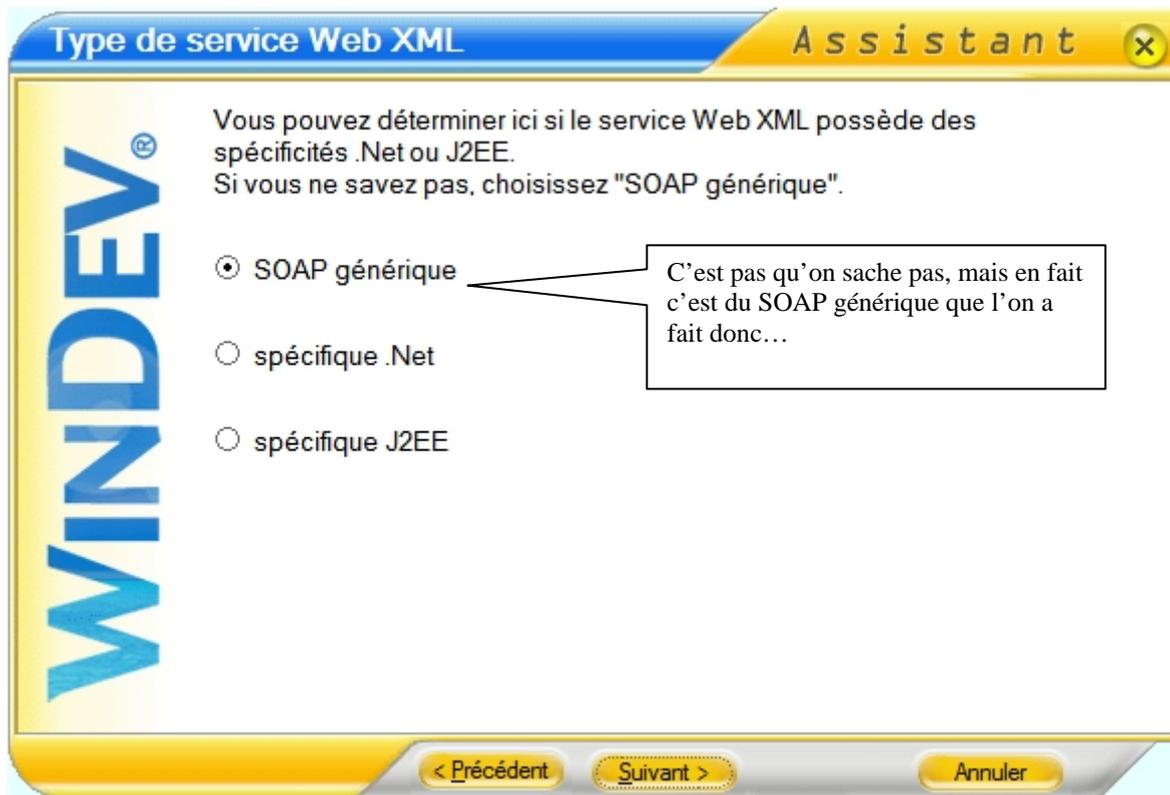
Nous allons maintenant intégrer la définition du service Web à l'intérieur de ce projet, pour cela :



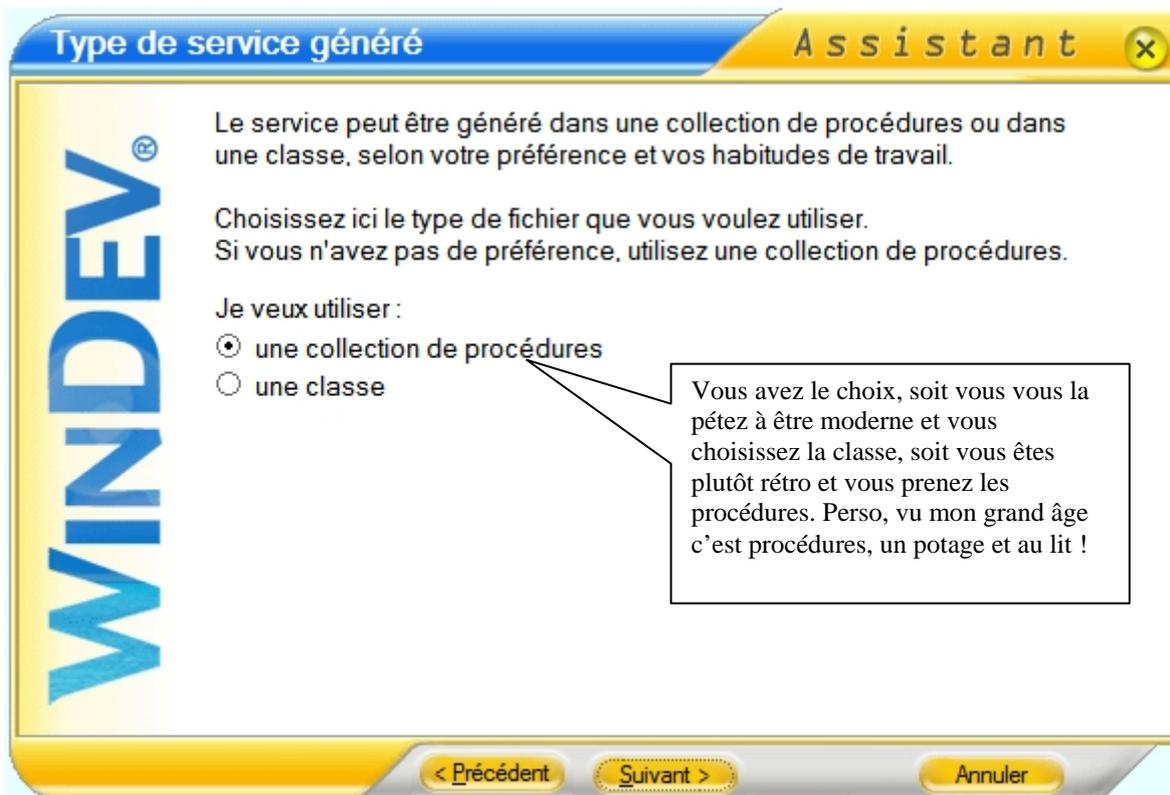
L'assistant d'importation se lance par la fenêtre suivante :



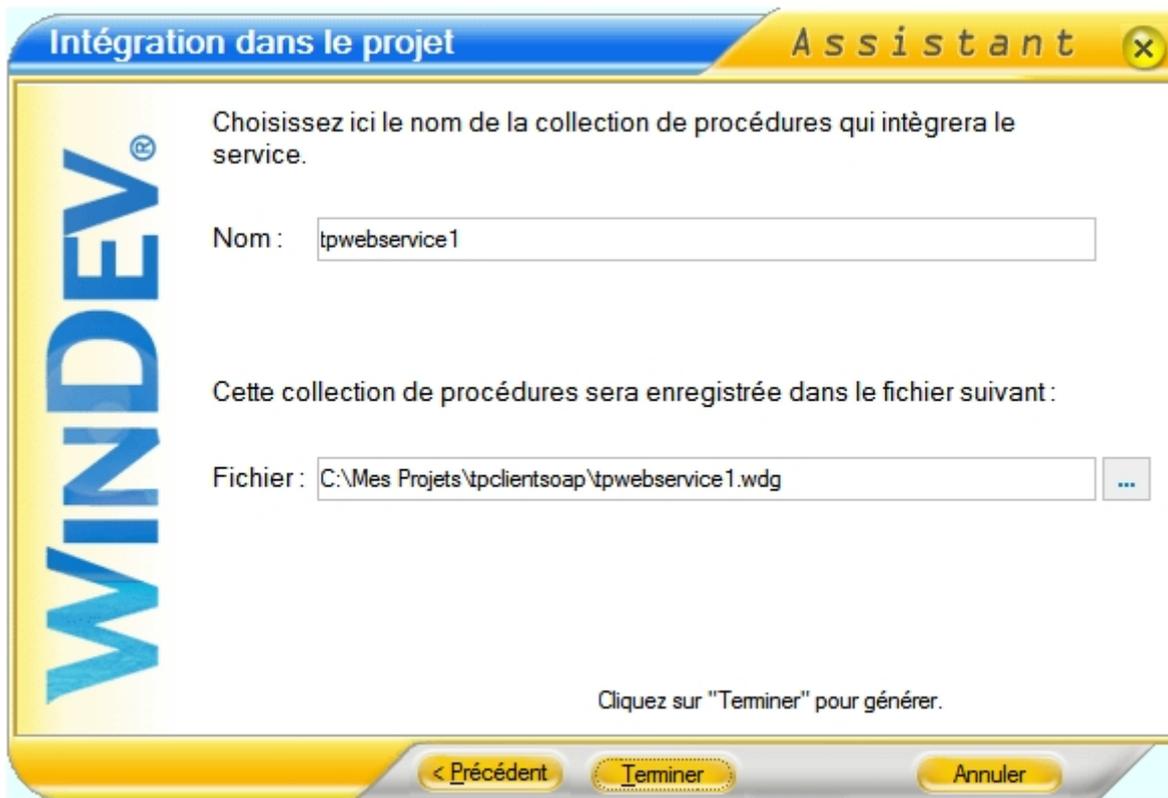
Les renseignements donnés on clique sur suivant



Aller, hop, un petit coup sur suivant... ! C'est lassant je vous dis !

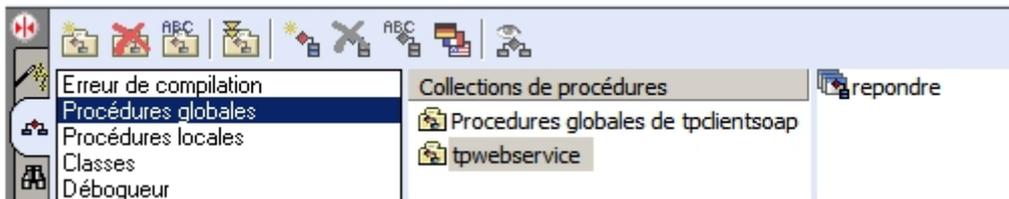


Vous avez deviné ? un ch'ti suivant !



Logiquement vous n'avez qu'à valider en cliquant sur terminer. C'est super les assistants, seulement gardez à l'esprit ce qu'ils viennent de nous faire réaliser et vous vous rendrez compte d'un sacré coup de main.

Un petit coup d'œil en bas de l'éditeur vous permettra de découvrir l'intégration de la méthode (dans le cas ou vous avez choisis d'intégrer le Webservice en temps que collection de procédure, sinon cherchez dans les classes)



Maintenant nous allons donner vie à tout ça, pour commencer créez un nouvelle fenêtre nommée départ qui sera la première fenêtre du projet. Elle va ressembler à ceci :



Ben, oui c'est tout !

Voici le code du bouton :

```
Info(repondre)
```

Explications : Vous vous rappelez que la procédure « repondre » du Webservice n'effectue qu'un renvoi d'une chaîne de texte contenant « coucou ». Donc logiquement en cliquant sur le bouton de la fenêtre nous devrions avoir une fenêtre d'information contenant la chaîne « coucou ».

Il n'y a qu'à essayer.

Lancez l'exécution du projet ou de la fenêtre, cliquez sur « Bouton » et si « coucou » apparaît vous pouvez prendre une pause, sinon...refaites l'exercice.

Maintenant allons visiter la plomberie, ouvrez le code de la procédure répondre, vous devez avoir ceci

```
PROCEDURE répondre()
bRes est un booléen

bRes=SOAPExécute("http://localhost/service.soap", "répondre", "tpwebservice", "tpwebservice/répondre")

SI PAS bRes ALORS
    SI SOAPErreur(SOAPErrMessage)~="" ALORS
        Erreur(ErreurInfo())
    SINON
        Erreur(SOAPErreur(SOAPErrMessage))
    FIN
FIN

RENVOYER SOAPDonneRésultat(SOAPRésultat)
```

```
bRes=SOAPExécute("http://localhost/service.soap", "répondre", "tpwebservice", "tpwebservice/répondre")
```

Cette ligne se charge de faire exécuter la procédure sur le serveur SOAP et renvoie le résultat dans une variable booléenne (Vrai si la communication avec le serveur SOAP a été établie, Faux dans le cas contraire).

```
SI PAS bRes ALORS
```

Tournure élégante pour dire si c'est pas vrai alors traitement de l'erreur.

```
RENVOYER SOAPDonneRésultat(SOAPRésultat)
```

La procédure a fonctionnée on retourne le résultat.