COURS WINDEV NUMERO 7

14/02/2015

Création d'un Web Service

Paramétrage d'un serveur Web,

Création du Service Web,

Création du client consommateur,

Approche XML, SOAP

…

Cours Windev Numéro 7

VERSION 19

Outils :

Un serveur d'application

Ce TP a été créé avec WinDev 19

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 qu'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.

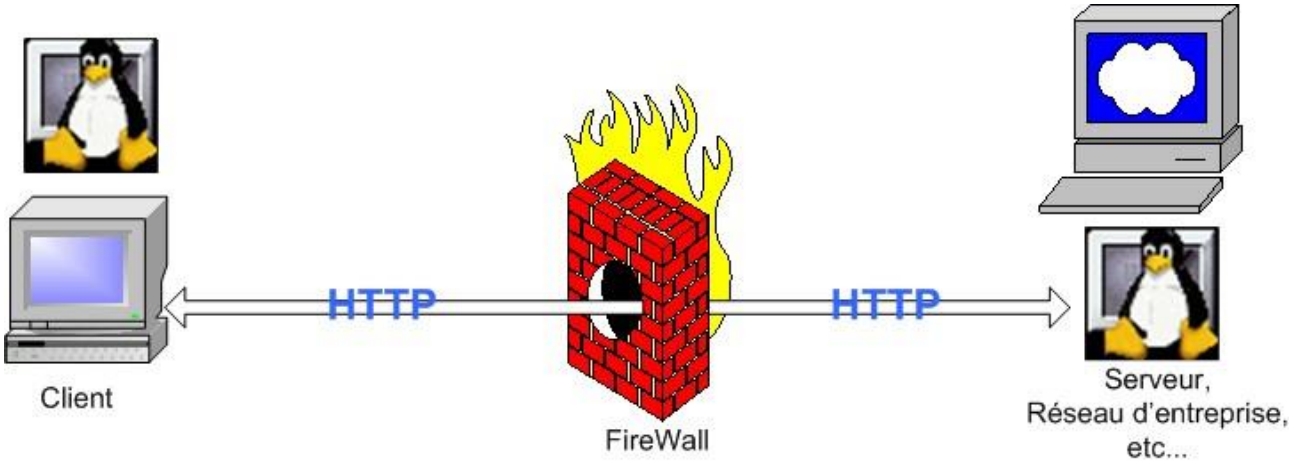
En effet, les WebServices se reposent sur des protocoles tels que XML et http, donc SOAP. Pour vulgariser ce dernier protocole, SOAP permet de faire circuler du XML via du HTTP. Donc lorsqu'on interroge un WebService, les données sont transmises en XML via le port 80 (HTTP). Rien de plus simple ensuite 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, **Dot Net, 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 ...



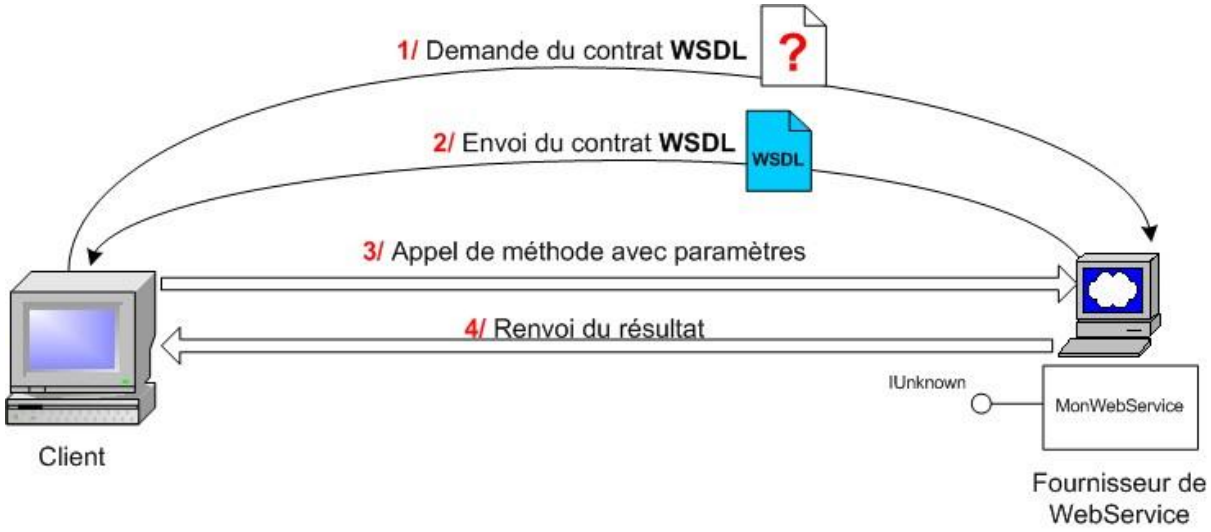
L'utilisation d'un WebService peut se diviser en différentes étapes :

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.

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.

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.

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 énumeration	✓	✓	✓
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 l. l 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 par 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 vais partir du postulat que vous avez un serveur web en ligne.

Vous pouvez créer une machine virtuelle avec une version de Windows et dessus y installer le serveur d'application WebDev - 10 connexions

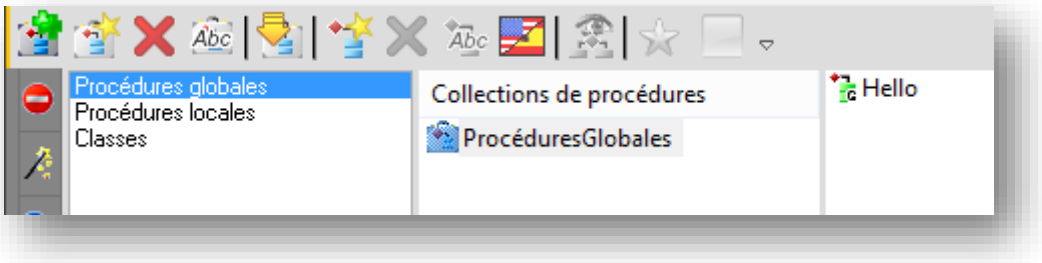
CREATION DU WEB SERVICE QUI SERA CONSOMME (COTE SERVEUR)

Bon, on y va ?

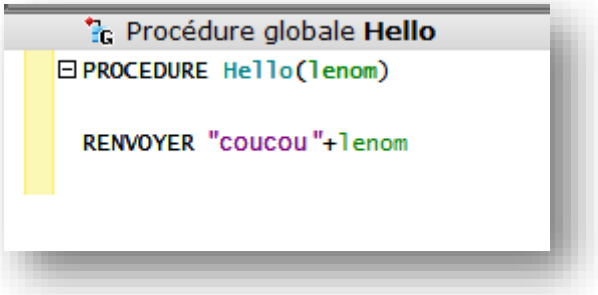
Ok, c'est parti !

Créez un nouveau projet : Webservice - Serveur. Ce projet ne contiendra aucune analyse, ni aucune fenêtre. 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 « Hello »



Le code est le suivant :



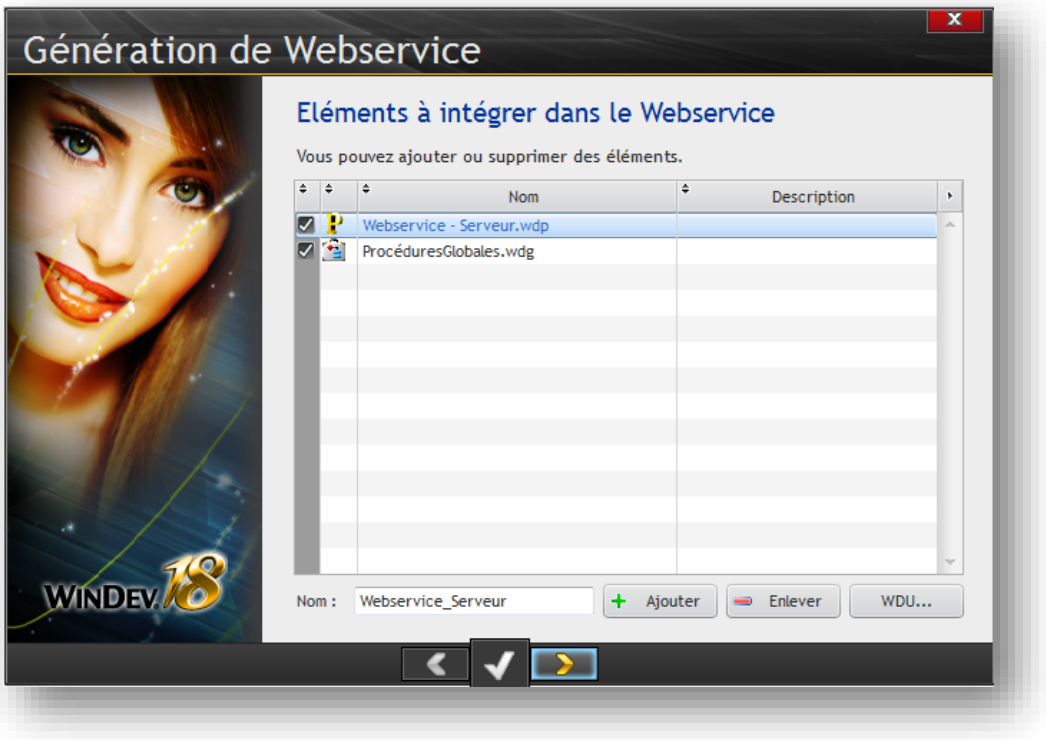
Comme vous le voyez : on se fatigue ! En fait pour ce premier exercice on se contente d'illustrer les principes donc je vous prie de m'excuser pour la faiblesse de l'illustration !!! Vous ferez mieux plus tard ;-)))

Vous venez de le comprendre que lorsqu'un client consommateur appellera la méthode « Hello » du WebService en passant un paramètre (lenom) il aura une chaîne de caractère « Coucou » + Lenom comme réponse. C'est puissant, non ?!!!

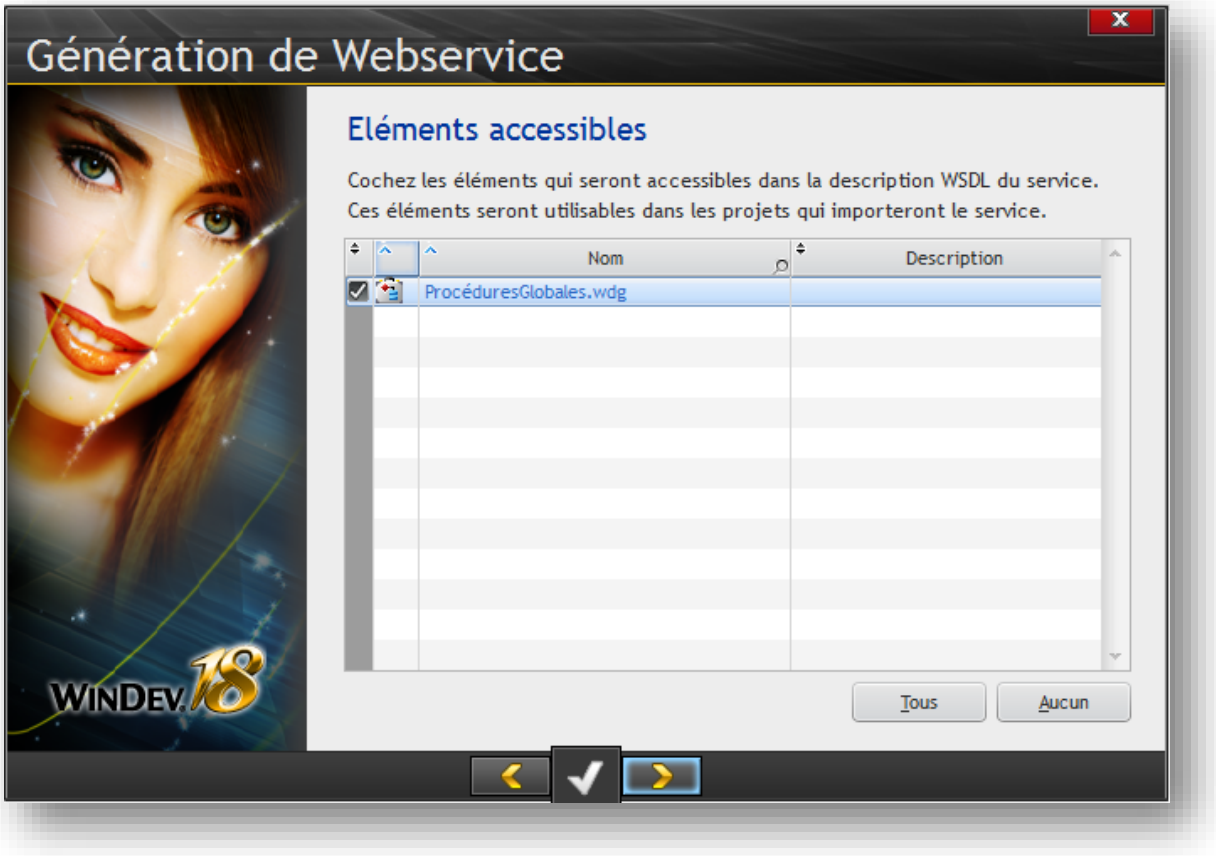
Bon, si je vous disais que le WebService est fini, vous le croiriez ? et pourtant c'est vrai !. Ah non, il reste à le déployer, on va le faire de suite :

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

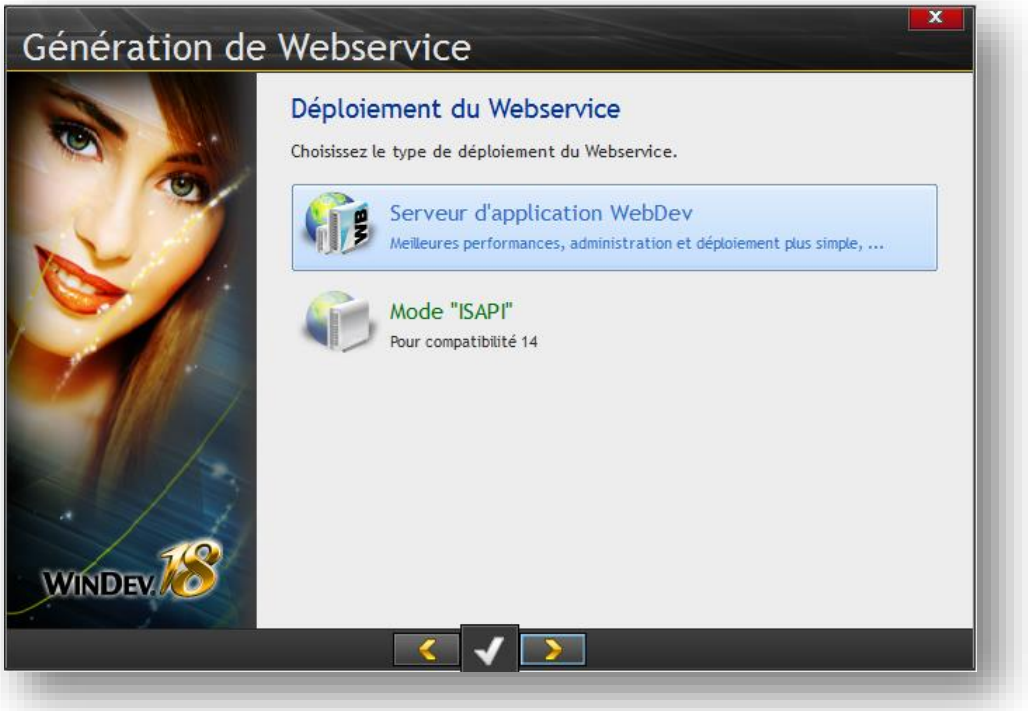
L'assistant se lance:



Sélectionnez les éléments comme ci-dessus et cliquez sur le bouton suivant.

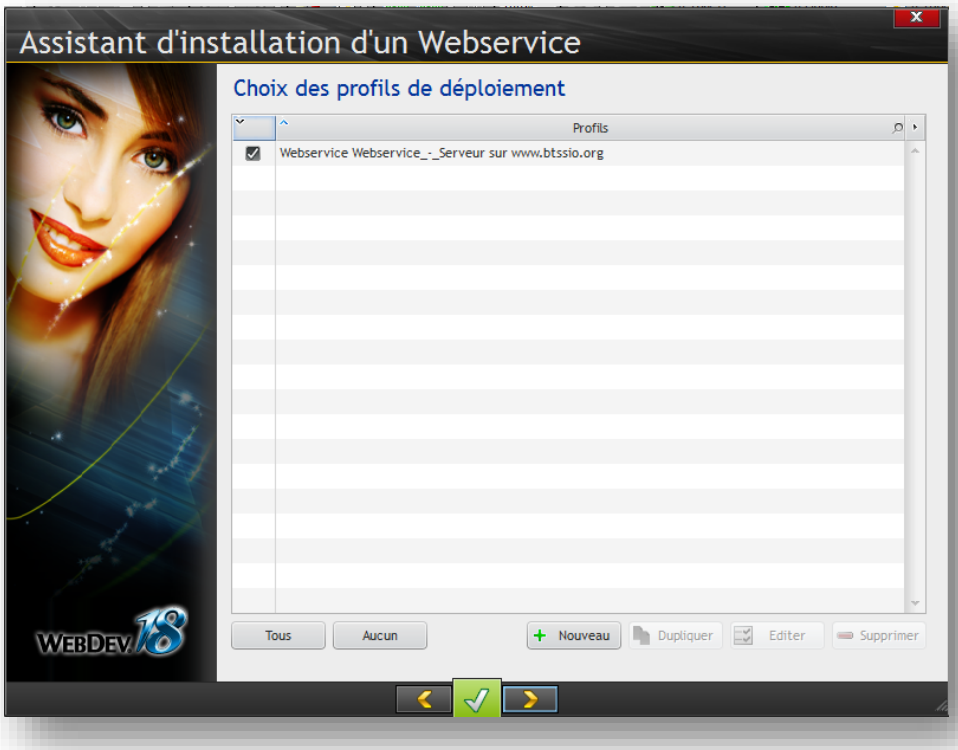


Cliquez sur suivant.

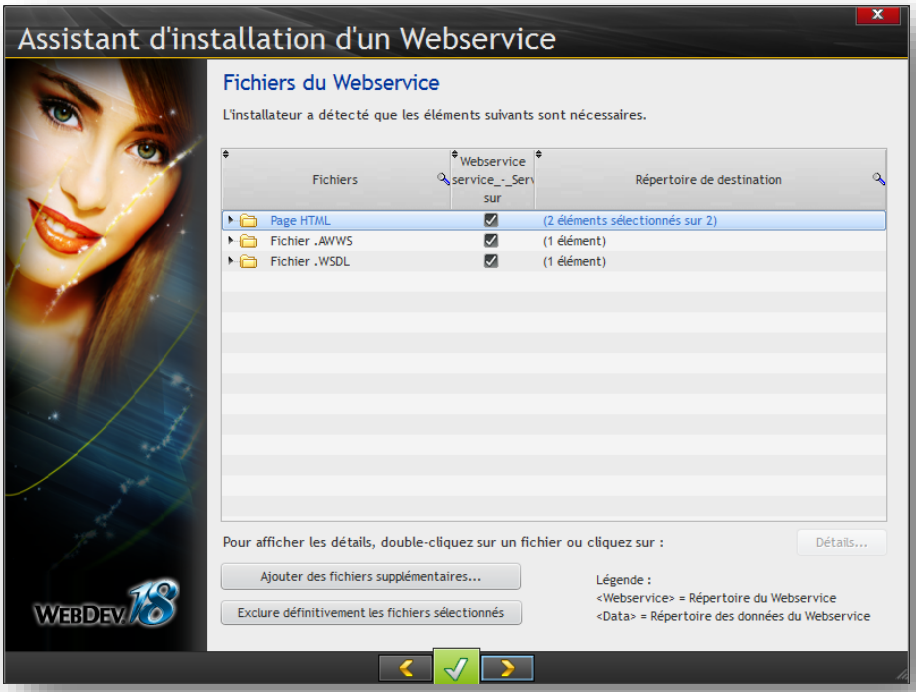


Ca deviendrait presque lassant ! Cliquez encore sur suivant.

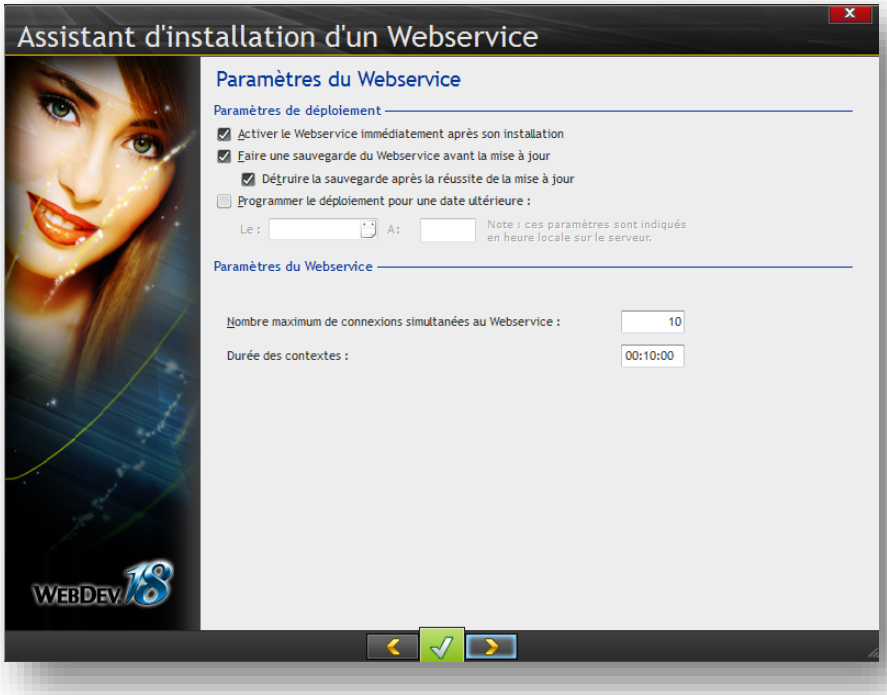
On continue à cliquer sur suivant



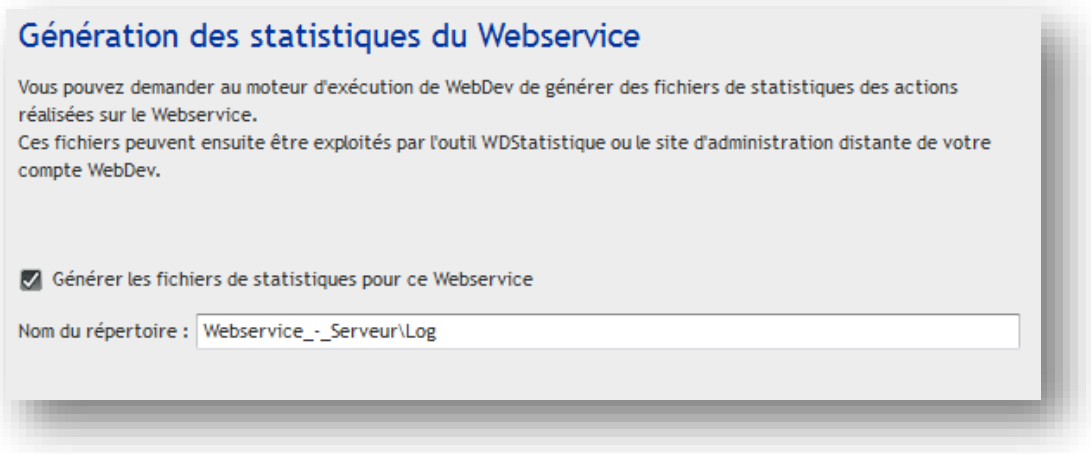
Comme vous le voyez, pour des raisons de test, j'ai déjà implémenté une installation sur www.btssio.org. Vous risquez de ne pas un affichage identique. Je vous laisse configurer le profil de déploiement et vous attends un peu plus loin !



L'installateur vous propose de sélectionner les éléments du projet, ne changez rien et passez à l'écran suivant.

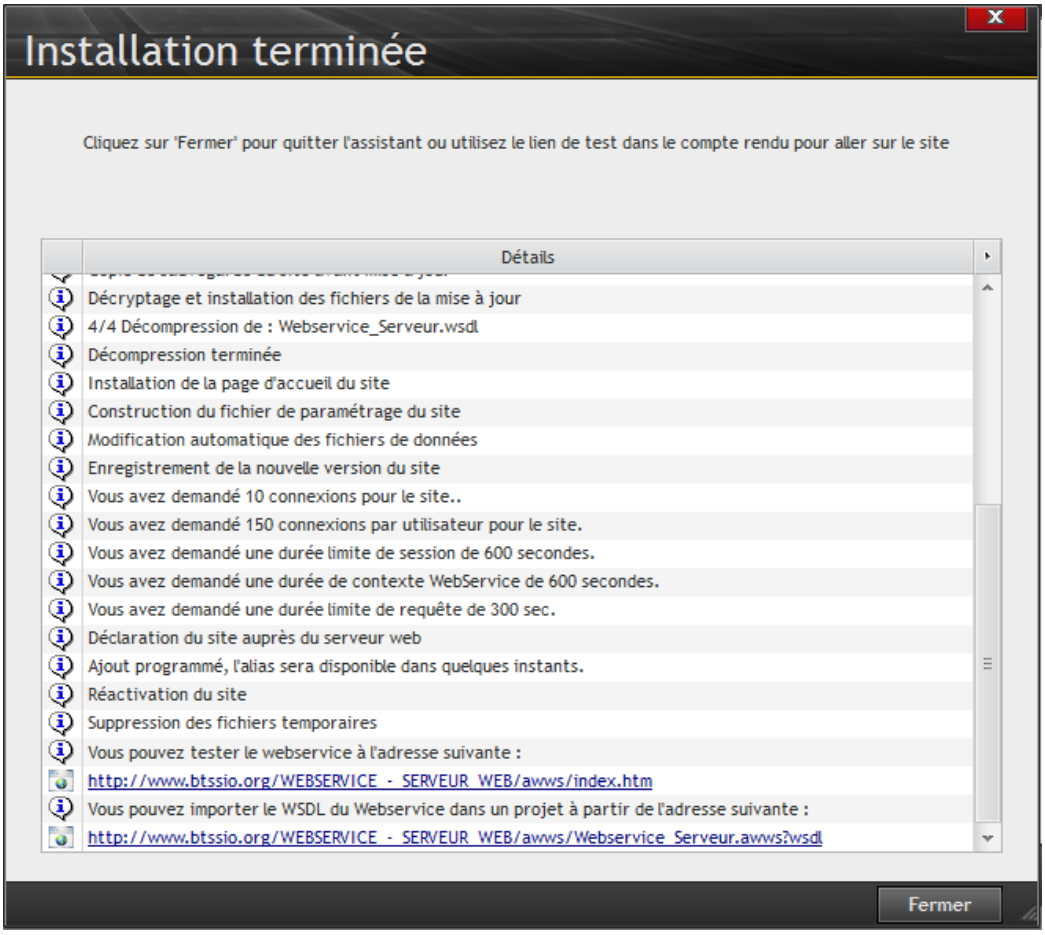


Comme vous pouvez le lire le Web Service est prêt à être déployer, passons à l'écran suivant.



Ici, vous indiquez où sera le fichier de log pour analyser les fréquentations (ou consommations) de votre Web Service.

Sur l'écran suivant on vous indique que l'assistant à toutes les informations pour clôturer la création du Web Service. Cliquez donc sur **Terminer.**



Et voilà ! C'est fini. Le web Service est terminé et installé sur un serveur Web. Il est donc prêt à être consommé !

Fermez le projet, nous allons passer à l'étape 2, la création du programme consommateur de Web Service

Prenez 5 minutes de pause, la cafetière fume !

CONSOMMATION DU WEB SERVICE.

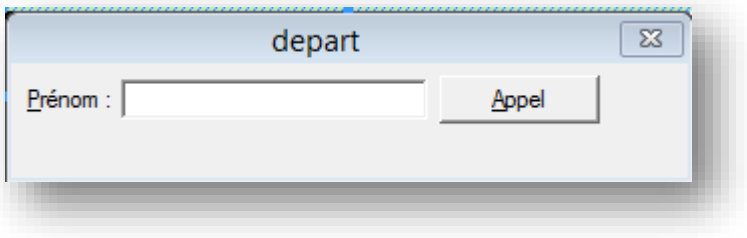
Maintenant, le morceau de bravoure : le client ou consommateur du WebService.

Nous allons créer le client.

Fichier ->Nouveau ->Projet.

Vous nommerez ce projet **WebService-Client**. Il n'utilisera pas d'analyse.

Créez une fenêtre nommée **Départ** qui ressemblera à ceci :



Elle est hyper minimaliste, mais va nous permettre d'illustrer la consommation du Web Service.

Nommez le champ texte Prénom : **Sprénom**

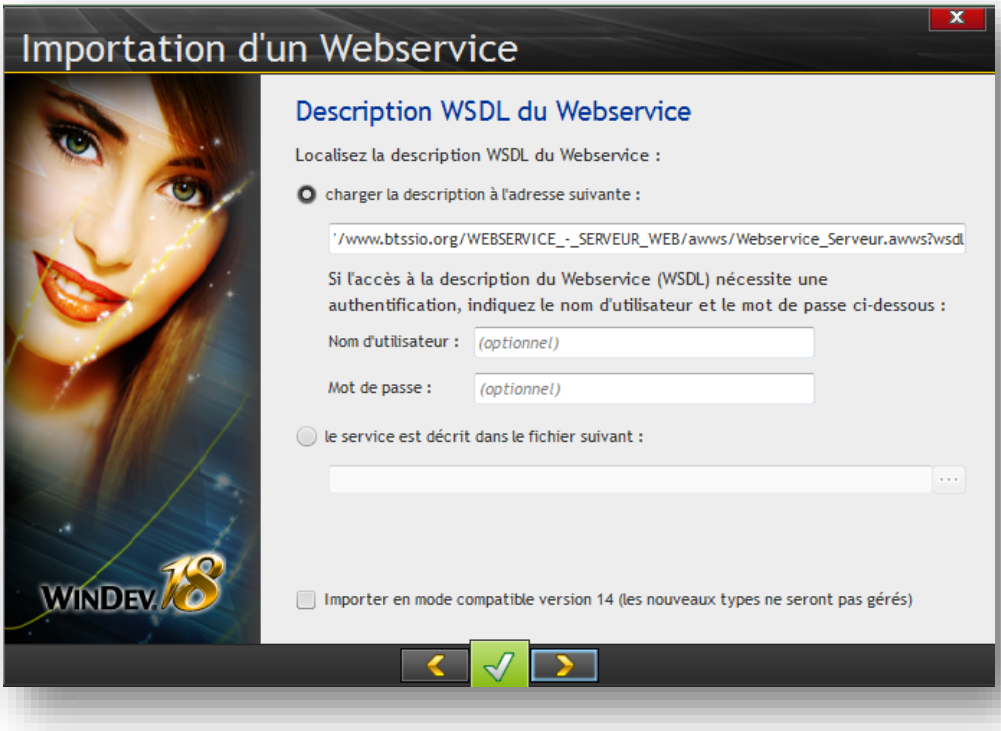
Nommez le bouton : **appe**l

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

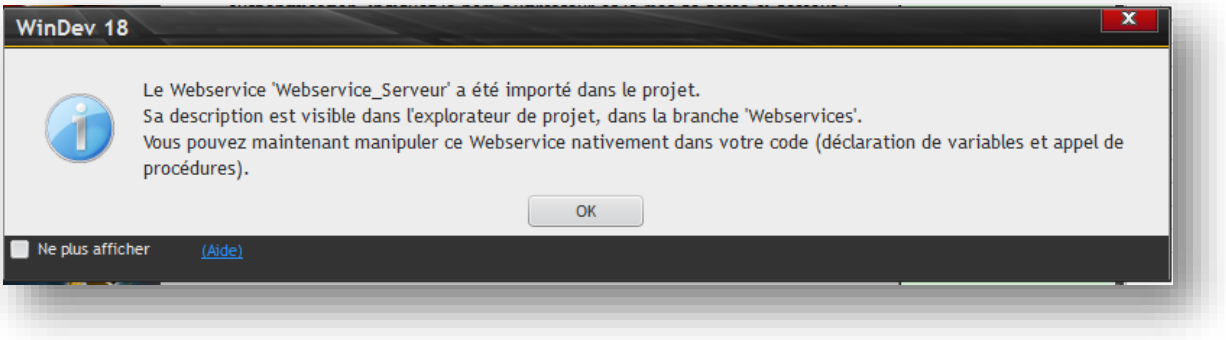
Cliquez sur le menu Atelier puis **WebServices (SOAP, .net, J2EE)**, puis sur **Importer et utiliser un WebService dans ce projet…**

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

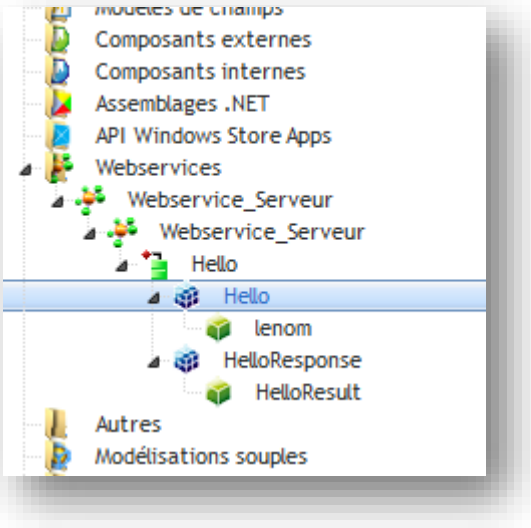
Cliquez sur suivant.



On fait pointer le chargement de la description sur l'adresse du serveur web sur lequel nous avons installé le Web Service et ensuite cliquez sur suivant



Si tout se passe bien vous devriez voir ceci.



Un petit coup dans l'explorateur de projet vous permettra de découvrir l'intégration du Web Service.

Maintenant nous allons donner vie à tout ça, voici le code du bouton appel:

Explications :

Vous vous rappelez que la procédure « Hello » du WebService n'effectue qu'un renvoi d'une chaîne texte contenant « coucou » + le Prénom passé en paramètre. Donc logiquement en cliquant sur le bouton de la fenêtre nous devrions avoir une fenêtre d'information contenant la chaîne « coucou » + Le prénom

Dans le code nous initialisons 2 variables :

Une nommée **reponse** est de type **Webservice_Serveur.HelloResponse**. Elle contiendra le retour du Web Service.

Une autre nommé parametre qui est du type **Webservice_Serveur.Hello**. Elle contiendra le paramètre à envoyer.

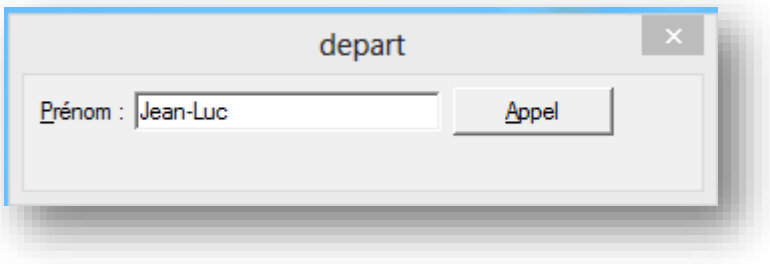
parametre.lenom=SPrénom contiendra le prénom saisi dans la zone de texte SPrénom.

reponse=Webservice_Serveur.Hello(parametre). Ici, le type réponse va recevoir le retour du web service.

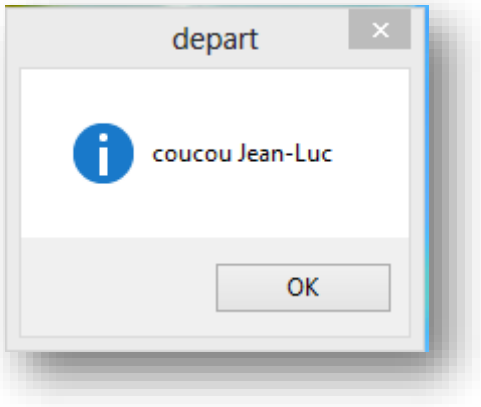
Info(reponse.HelloResult). Maintenant, nous faisons afficher la zone **HelloResult** du type **reponse**.

Ça va ? Vous n'êtes pas au fond du trou ?

Bon alors testons !



Si vous cliquez sur le bouton Appel, vous devriez voir ceci :



Et voilà ! Le tour est joué ! Vous venez de façon simple et ludique de créer et manipuler des web services ! Trop fort !