Comment appeler un WebService depuis une application générée avec FormPublisher

1. Introduction

Ce tutorial-cookbook montre comment appeler un WebService simple, depuis une application générée avec FormPublisher. Par "simple", il est question d'un WebService qui prend quelques paramètres en entrée et qui fournit quelques données en résultats (à la différence d'un WebService plus sophistiqué qui fournirait des listes de réponses de tailles dynamiques).

L'exemple porte sur un service de conversion monétaire : à partir de deux devises, le système tiers fournit le taux de change.

Pour suivre ce cookbook, il faut au moins avoir suivi "Comment faire un formulaire".

2. Initialiser un nouveau projet/document

Rappel : les WebServices sont fournis par le module CommunicationPack.

Initialiser un nouveau projet/document pour cette démonstration ou ouvrir un projet/document existant.

Dans le document JForm, ajouter une section "Input" pour recevoir les paramètres et une section "Output" pour recevoir les résultats. Ces noms de Section restent au libre choix de l'auteur FormPublisher.

3. Recueillir les paramètres

Dans la Section "Input", placer deux champs pour le choix des devises :

Devise de départ	
Devise d'arrivée	

Ces champs portent ici les noms "deviseDepart" et "deviseArrivee".

Accessoirement, le premier est une ListBox et le second une ComboBox, mais cela n'a aucune importance dans le cadre de cet exemple de WebService.

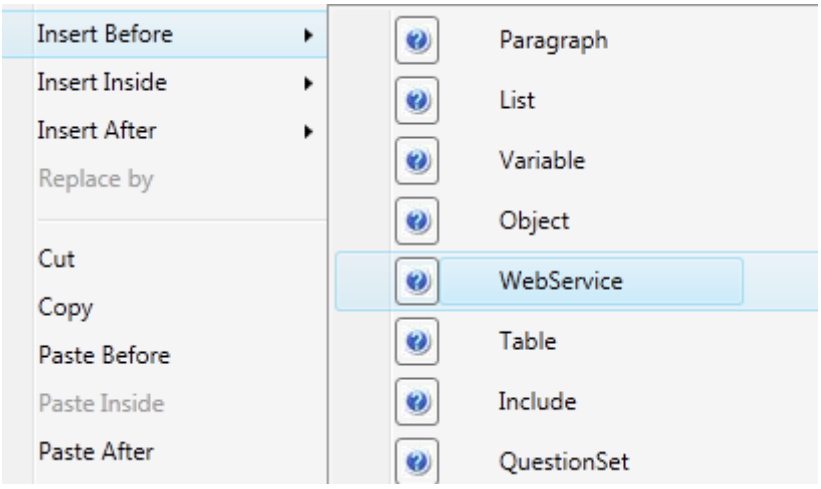
4. Préparer le résultat

Dans la Section "Output", placer un champ pour recevoir le résultat :

Taux de conversion / :	

5. Poser le WebService

Dans la Section "Output", avant le réceptacle du résultat, placer un élément WebService.

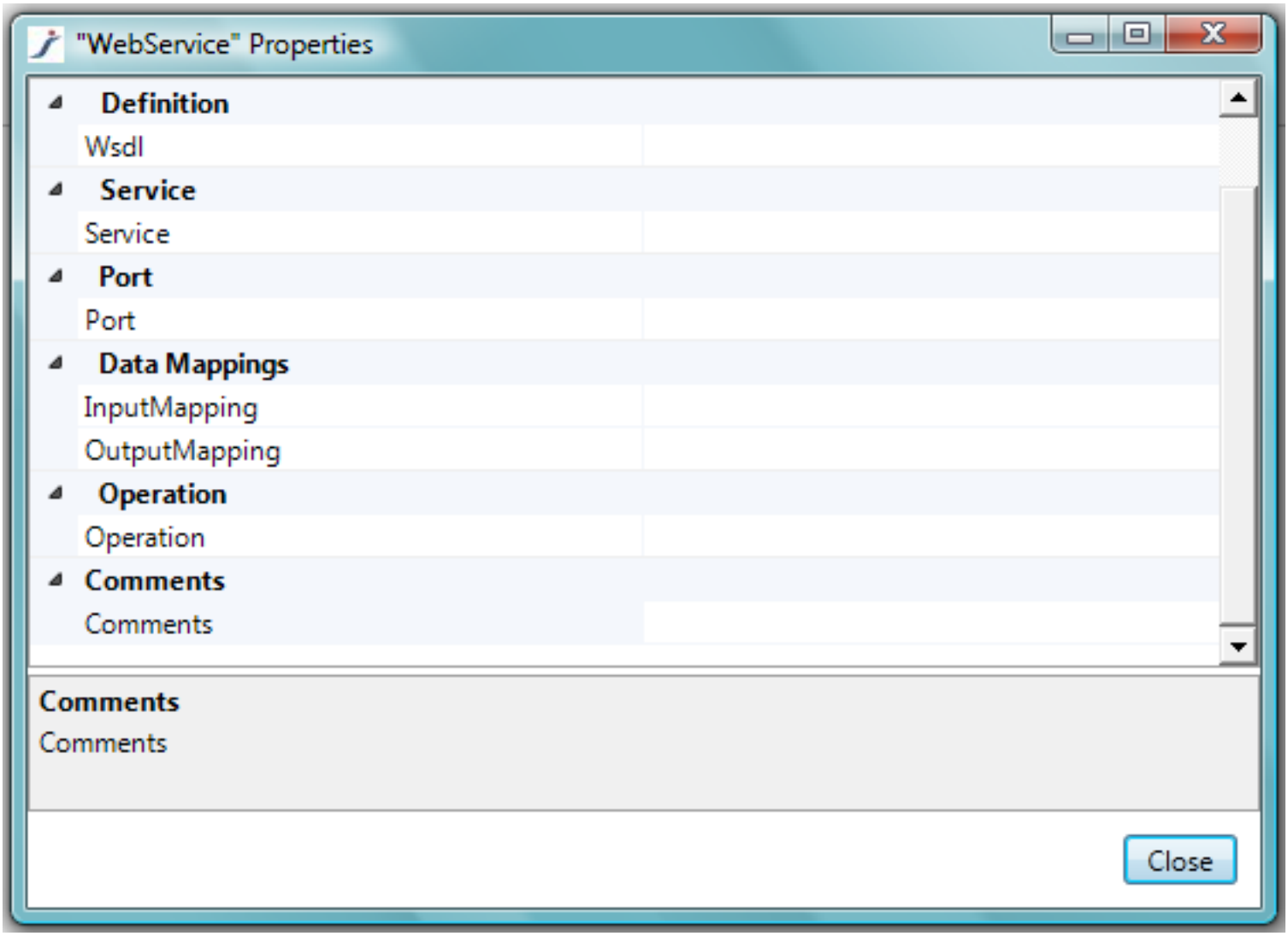


Cet élément WebService constitue le centre de cette démonstration. Il provoquera un appel au WebService tiers lors de l'affichage de la page sur laquelle il est posé.

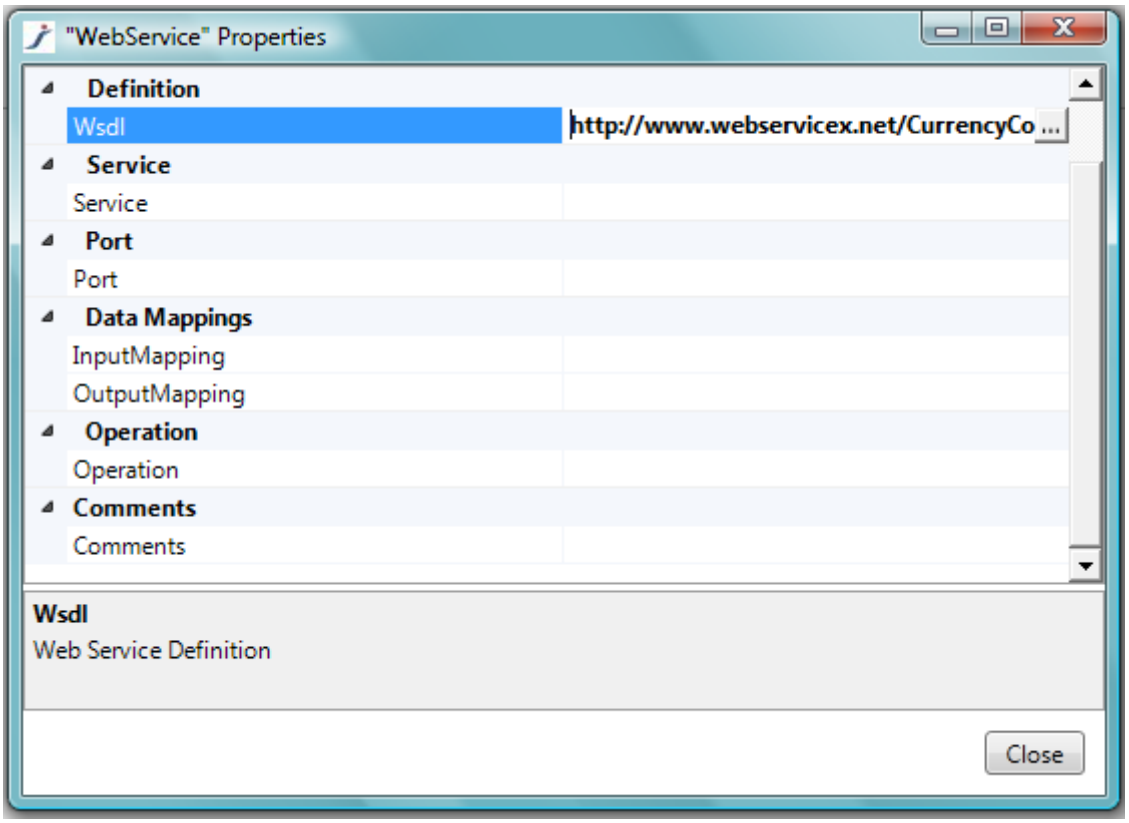
6. Pointer le WSDL

Un WebService expose son offre à ses consommateurs à travers un document WSDL (WebService Definition Language). Cette norme WSDL permet aux consommateurs du WebService de s'y "brancher" automatiquement. Le document WSDL peut être fourni sous la forme d'un fichier ou d'un lien web. Le Studio va extraire automatiquement les informations nécessaires publiées dans le WSDL.

Ouvrir les propriétés de l'élément WebService :



Pointer le WSDL :

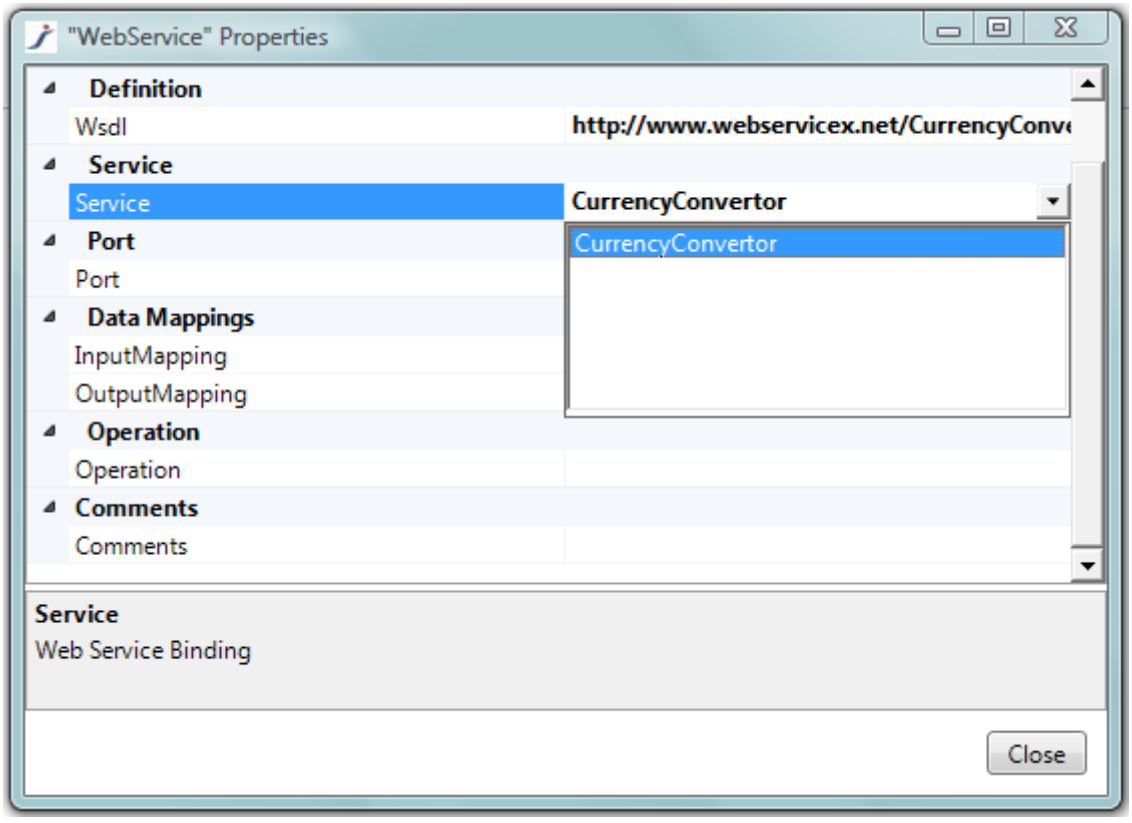


Par ce "branchement" direct du projet FormPublisher au WSDL du WebService, le projet va pouvoir à présent spécifier sa requête et ses paramètres.

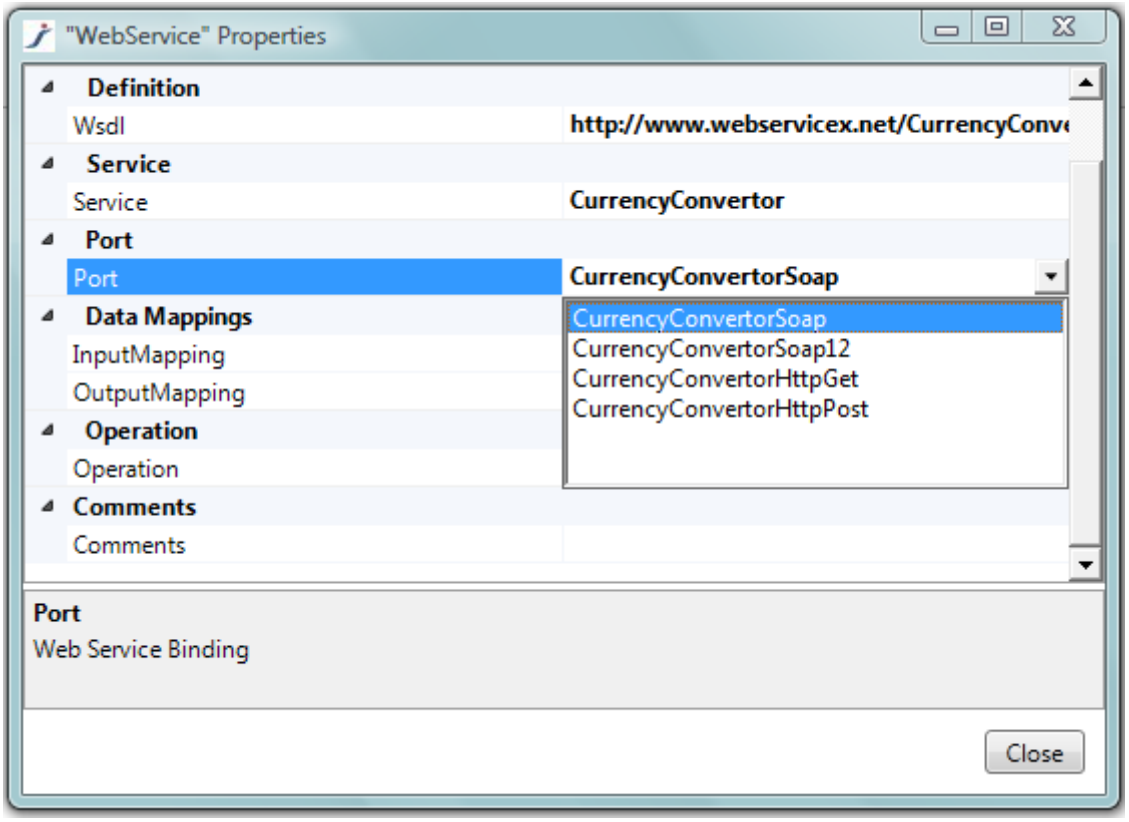
7. Sélectionner l'action

Comme une offre WSDL peut comporter plusieurs choses, les propriétés "Service", "Port" et "Operation" servent à sélectionner l'action retenue (ici, l'obtention d'un taux de conversion monétaire).

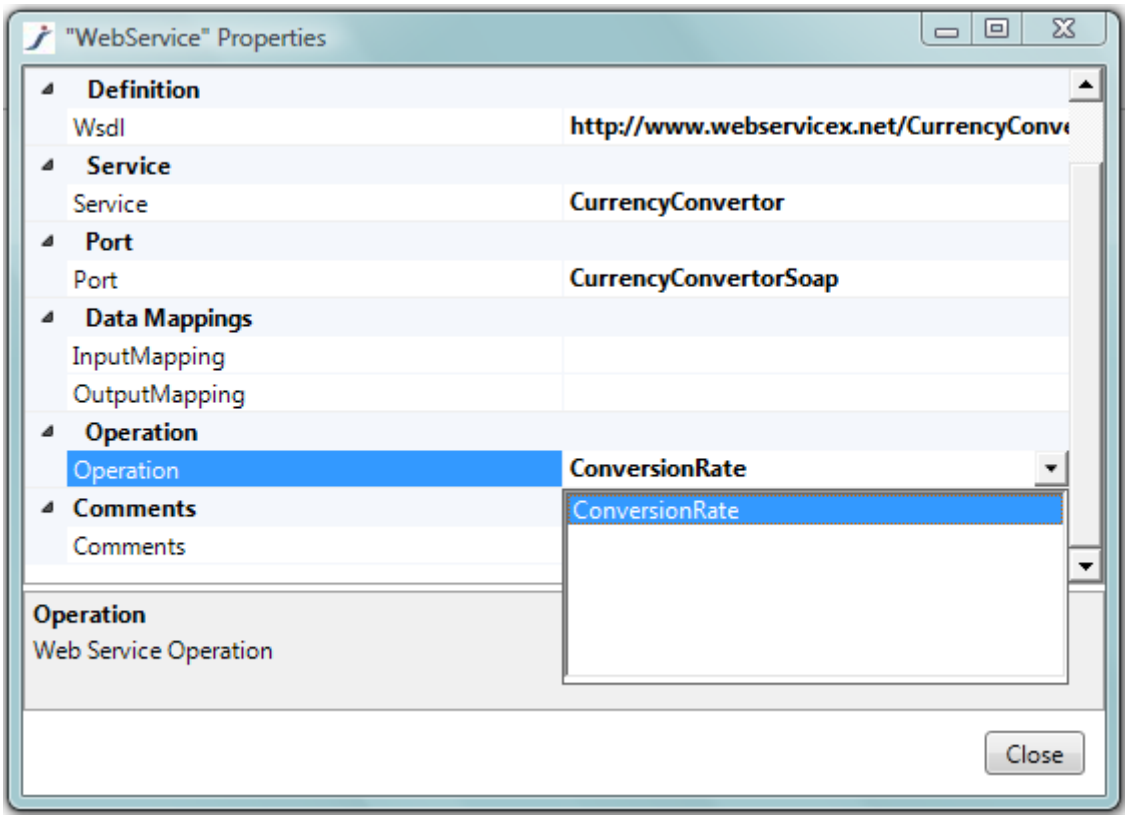
Sélectionner le "Service" :



Sélectionner le "Port" :



Sélectionner l'"Opération" :

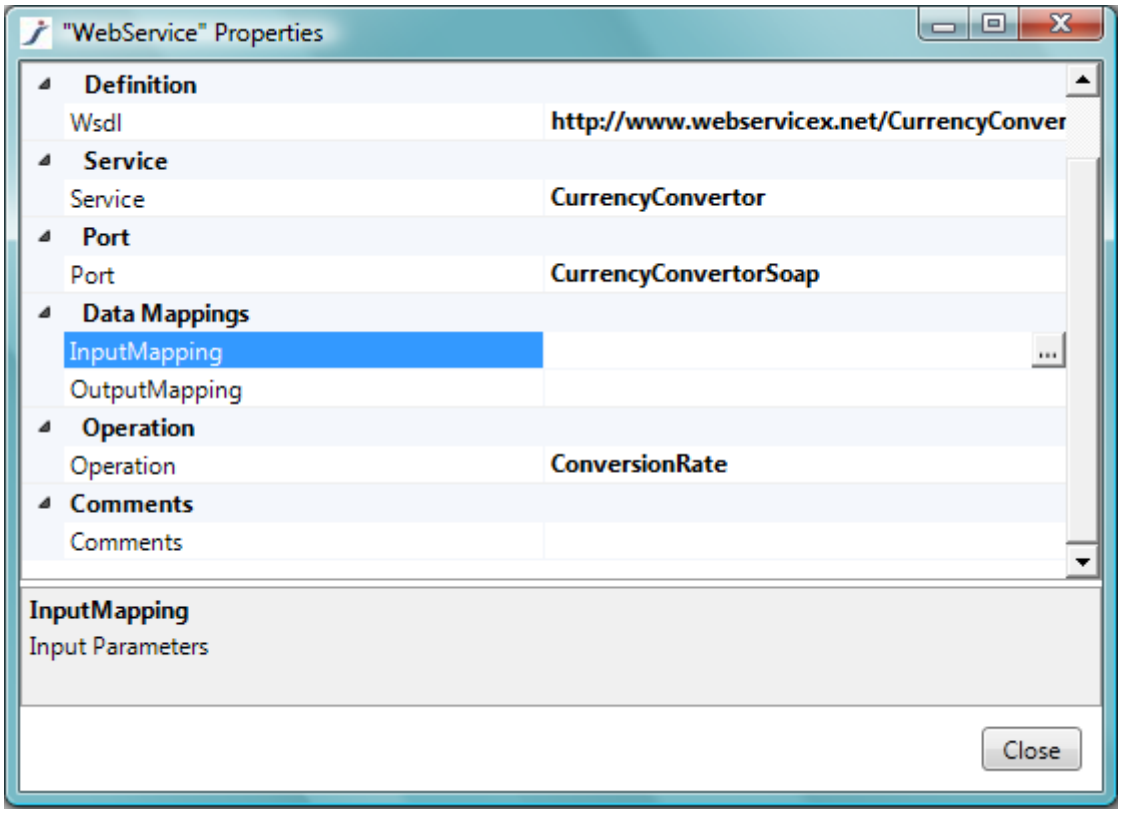


A l'issue de ces sélections, l'action du WebService est définie et le paramétrage va pouvoir être réalisé.

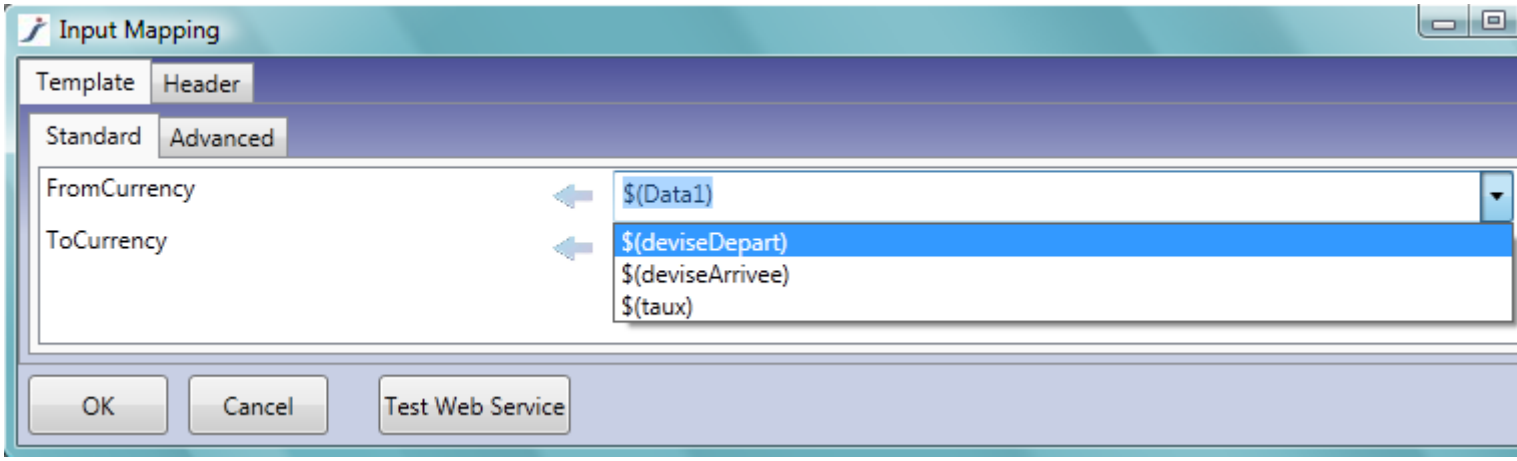
8. Brancher les paramètres

La mise en correspondance entre les champs de saisie du formulaire et les paramètres d'appel du WebService est appelée "Input Mapping".

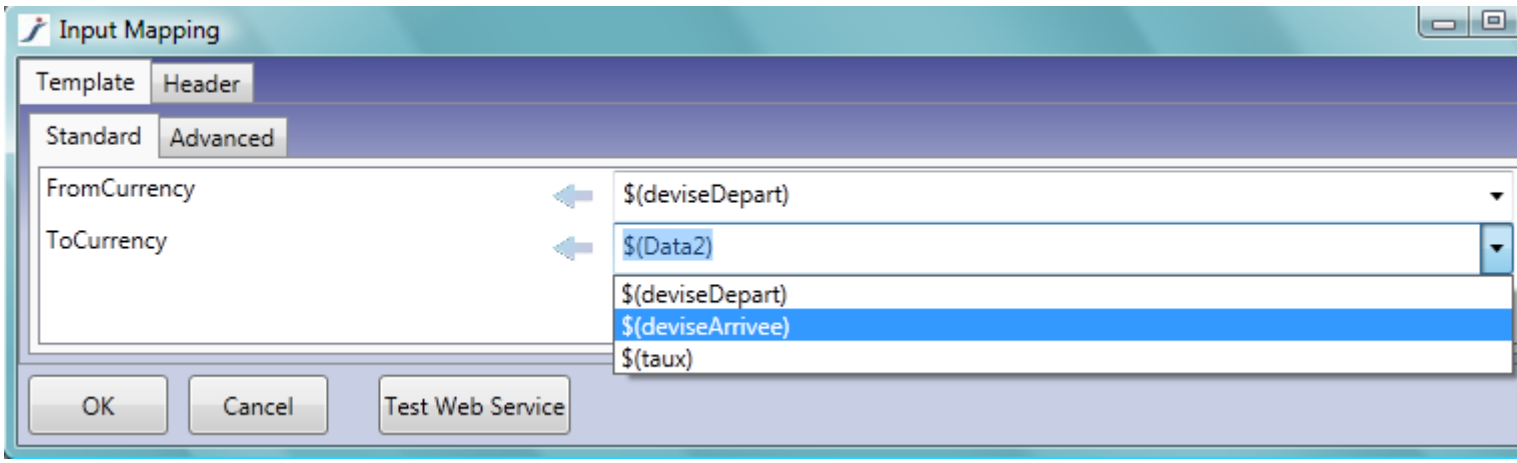
Ouvrir les propriétés du WebService et sélectionner "InputMapping" :



L'assistant de mise en correspondance s'ouvre pour relier aisément les champs de saisie du formulaire aux paramètres attendus par le WebService.

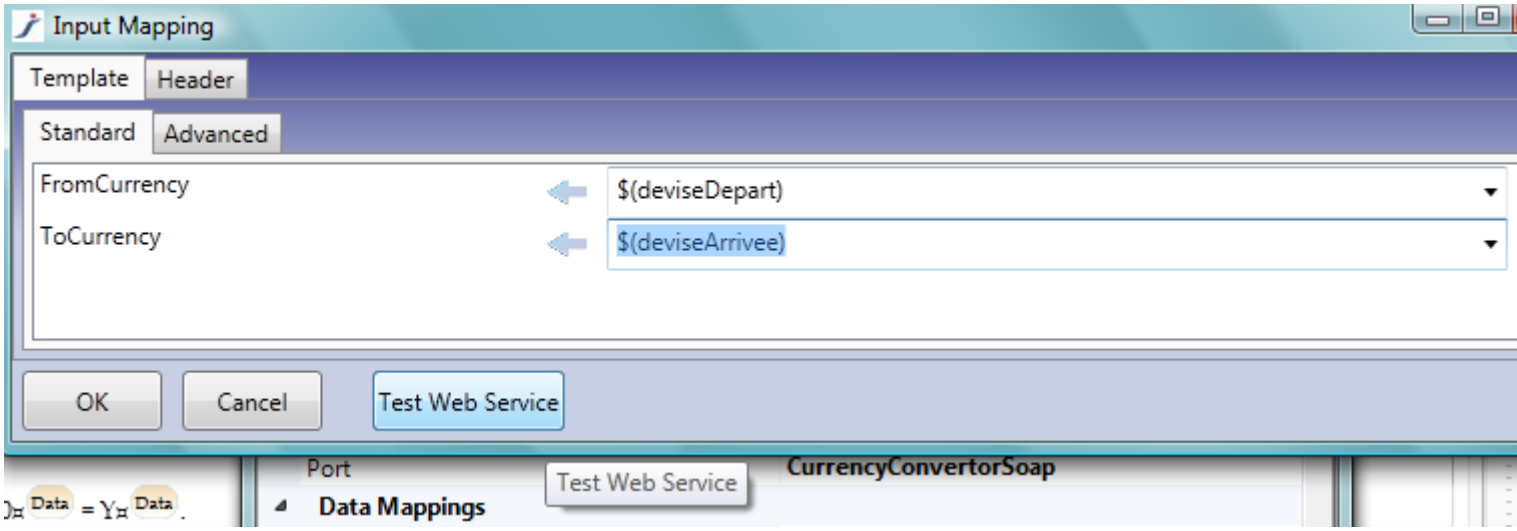


A gauche figurent les paramètres attendus, à droite, des listes déroulantes alimentées par le dictionnaire de données du formulaire. Remplacer le premier paramètre par le champ "deviseDepart" et le second par "deviseArrivee".

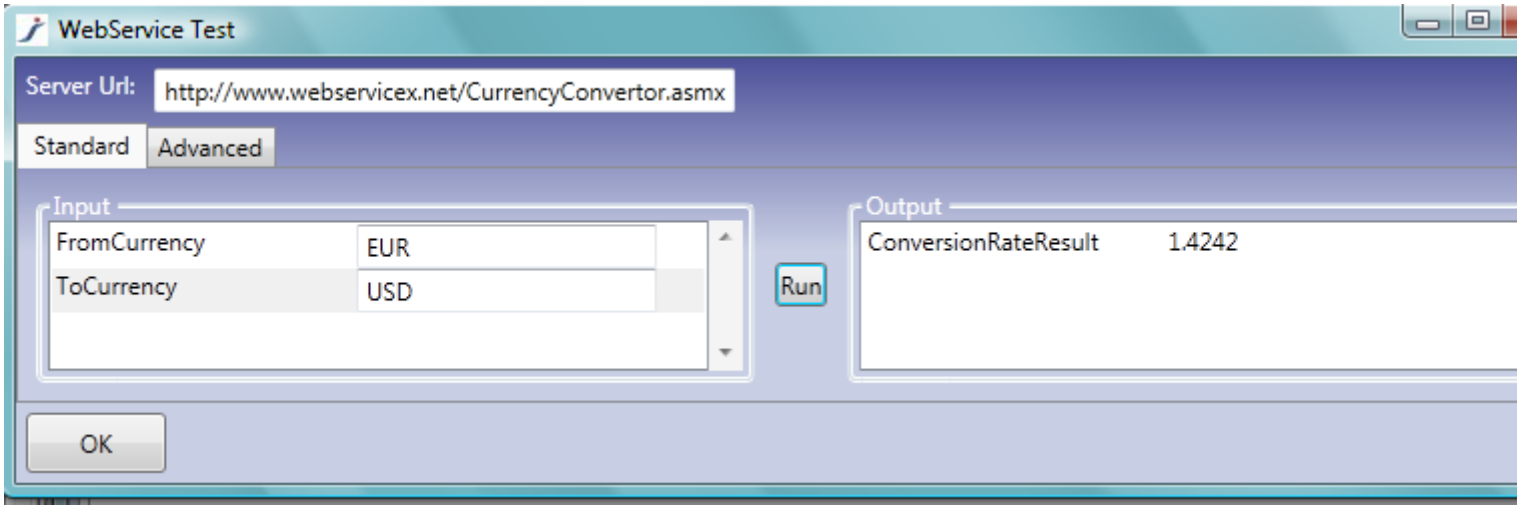


9. Tester les paramètres

Le Studio permet de tester l'InputMapping du WebService directement depuis l'assistant, sans encore générer l'application.
Une fois les correspondances renseignées, cliquer sur "Test Web Service".



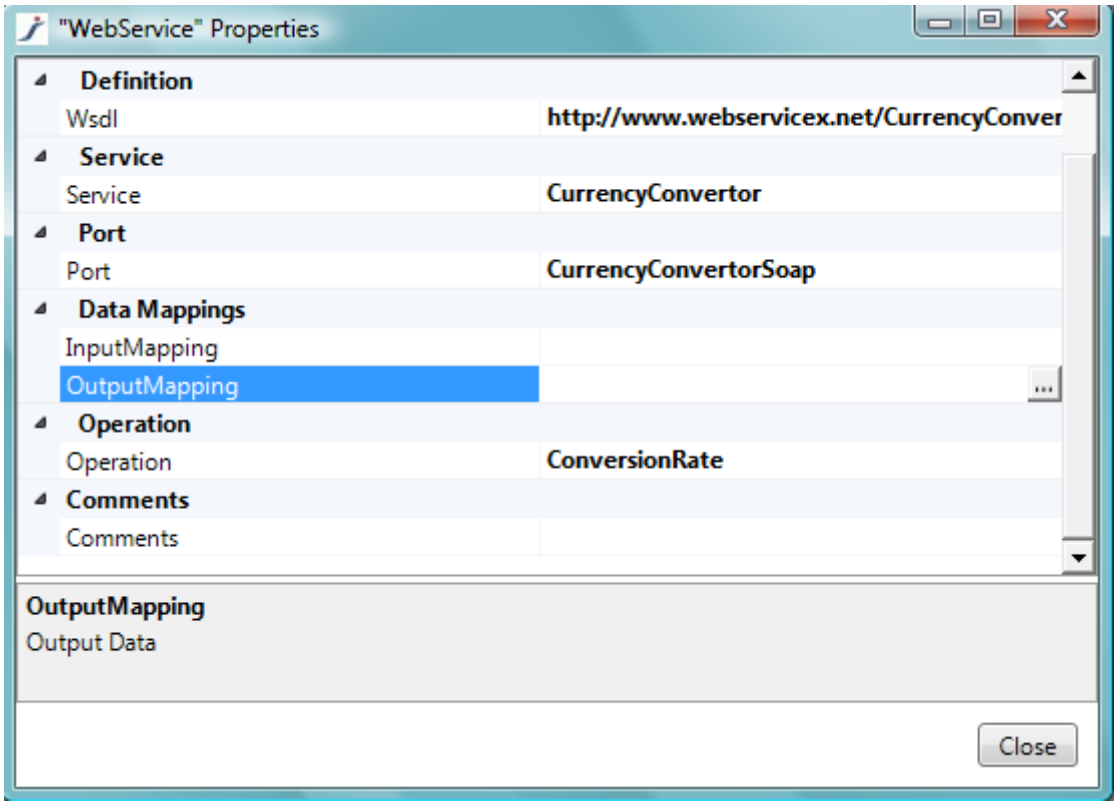
Un autre assistant apparaît pour tester le WebService depuis le Studio :



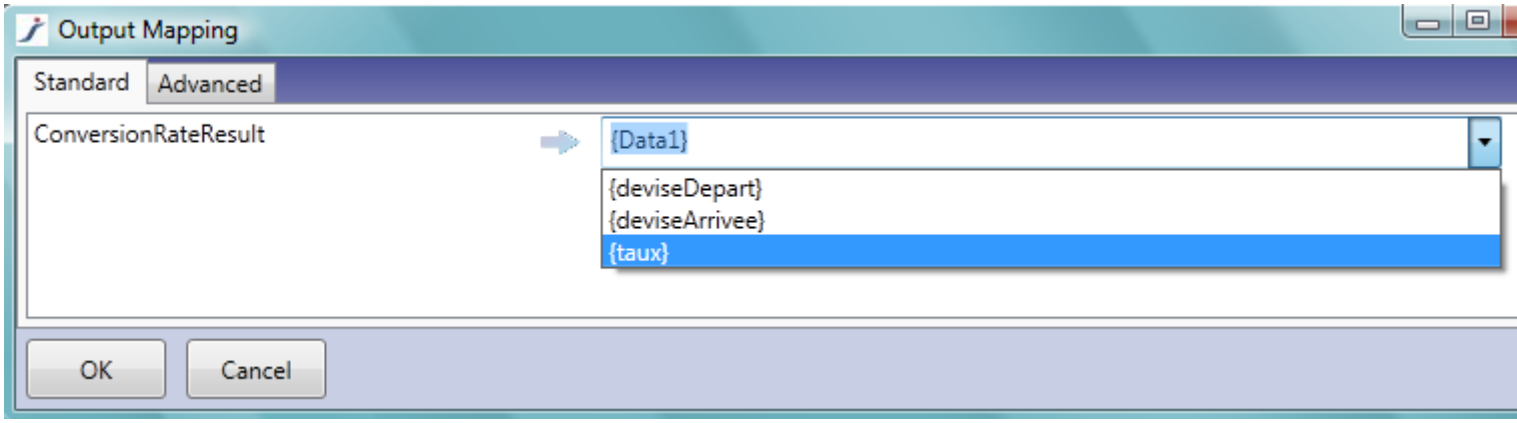
Saisir deux valeurs dans "Input", cliquer sur "Run" et constater le résultat dans "Output". Cet "Output" va maintenant être raccordé au réceptacle de résultat, comme l'ont été les champs de saisie aux paramètres.

10. Brancher le résultat

Ouvrir les propriétés du WebService pour sélectionner cette fois-ci l'Output Mapping :



L'assistant affiche à gauche les données renvoyées par le WebService et permet des mises en correspondance avec le dictionnaire de données via les listes déroulantes à droite :



Le paramétrage "in/out" du WebService est achevé.

11. Gérer les anomalies

A l'intérieur de l'élément WebService, il est possible de poser des OnErrorAction. Leur @ActionType définit le type d'anomalie (pas de réponse du tout ou réponse incorrecte). Leur @ErrorType définit le degré de gravité (simple avertissement ou erreur bloquante).

Chaque OnErrorAction peut contenir un OnErrorMessage pour renseigner l'utilisateur sur le déroulement des événements.

12. Générer et lancer l'application

Générer, lancer, tester, empaqueter, déployer l'application s'effectue comme à l'ordinaire par Build, Run, Package, etc. Un premier test d'intégration avec le WebService peut être fait depuis le Studio, via Build+Run. Un test plus proche de l'environnement de production sera obtenu par Build+Package, suivi du déploiement de l'application générée (fichier.war) sur un serveur de test. Si les essais sont concluants, l'application est déjà prête au passage en production.

13. Conclusion

En suivant pas à pas ce tutorial-cookbook, un utilisateur (non informaticien) formé à FormPublisher peut **concevoir et réaliser une application web qui utilise des WebServices** de type SOAP.

Le fournisseur du WebService n'a qu'à publier son offre au format standard WSDL. A partir du WSDL, le Studio automatise les branchements des paramètres et du résultat.

A l'élément WebService de FormPublisher sont adjoints OnErrorAction et OnErrorMessage pour traiter dynamiquement les éventuelles anomalies du WebService lors de l'exécution de l'application générée.