CHAPITRE 1

Introduction aux web services

Contenu du chapitre :

Env. De dev.	Langage
Visual Studio	Java EE
Qt Creator	C#
NetBeans	JavaScript
Eclipse	Objective C
Xcode	PHP
	HTML

Objectifs du chapitre :

Ce chapitre a comme ambition de proposer une définition des web services. Sans constituer un cours, il rassemble des pratiques et des considérations dont la connaissance est souhaitable pour créer et mieux utiliser des web services. De même, quelques notions sur les protocoles de communication sont rappelées, afin de mieux comprendre le déroulement des étapes lors de l'utilisation d'un environnement de développement tel que Visual Studio ou NetBeans.

Systèmes					
Windows	x	Linux	x	Mac	x

1.1 Définition

Il existe de nombreuses définitions d'un web service. Citons la définition de l'organisme de normalisation du World Wide Web Consortium (W3C) en anglais :

"A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards" (Source : http://www.w3.org/TR/ws-arch/).

Après traduction, on obtient la définition suivante :

"Un Web Service est un système logiciel conçu pour permettre l'interopérabilité entre les machines sur un réseau. Il possède une interface qui décrit, dans un format normalisé, le moyen de communiquer avec la machine (par exemple : WSDL). D'autres systèmes interagissent avec les web services, conformément à l'interface, en utilisant les messages SOAP envoyés par le protocole HTTP et écrits en XML, en liaison avec d'autres normes standards du Web."

Nous en proposons une autre, partielle aux yeux des spécialistes, mais suffisante dans ce livre :

"Un web service est une application informatique possédant une URI (Uniform Resource Identifier), hébergée par un serveur d'applications, qui est composée de procédures dont l'exécution représente un service proposé à un autre programme informatique (nommé client) et qui est accessible sur internet par l'utilisation de protocoles standards (HTML, XML,...)".

Les web services créent une architecture de type client/serveur dans laquelle des clients (ordinateur de bureau, ordinateur portable, téléphone portable…) utilisent, via internet, des procédures qui sont stockées sur un serveur d'applications (Figure 1-1).

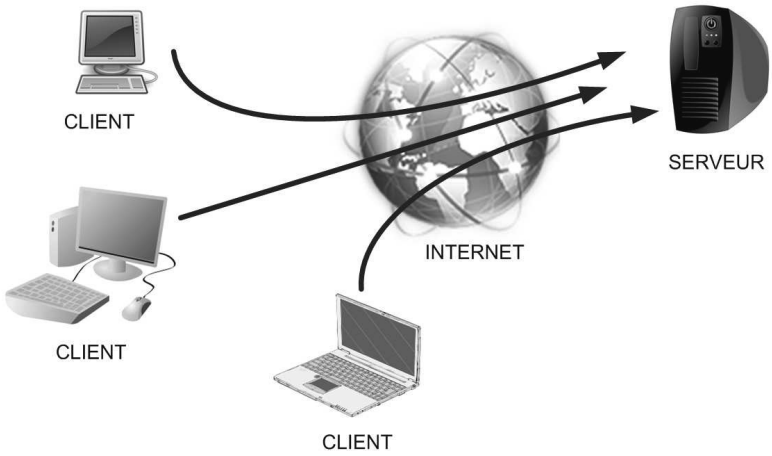

Figure 1-1. Architecture client/serveur

De manière schématique, trois composants (Figure 1-2) sont nécessaires dans un web service :

- Un protocole pour décrire le service (idéalement il doit lister les méthodes disponibles et leurs arguments…) ;
- Un protocole décrivant la composition des messages ;
- Un protocole de transport pour faire circuler les informations sur Internet.

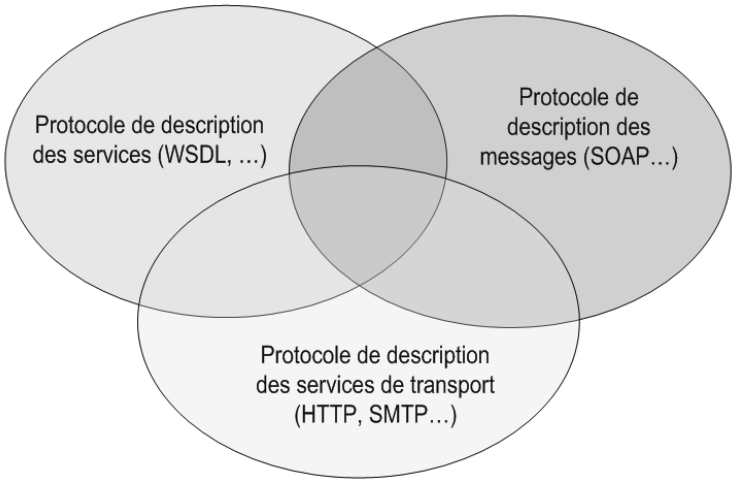

Figure 1-2. Les différents composants d'un web service

1.2 Les caractéristiques d'un web service

Il faut considérer deux points de vue complémentaires (parfaitement résumés dans le document proposé par [SMI 12]) :

- le point de vue de l'utilisateur (un programmeur ici) qui utilise une méthode proposée dans un web service pour la réalisation de l'un de ses programmes. Cet utilisateur est sensible aux temps de réponse, à la disponibilité du web service c'est-à-dire à des critères qualitatifs qui concernent la réalisation d'opérations déléguées aux web services. L'utilisateur est sensible à la **qualité de service** nommée aussi **performance du web service** (Figure 1-3).
  Souvent, le terme **capacité d'accueil** est utilisé pour désigner le nombre maximal de connexions autorisées sur le web service. Le mot autorisé s'entend ici, soit comme une limitation matérielle due à la capacité de la machine hôte, soit comme une limite imposée par les concepteurs du web service.
  Cette capacité acquiert une grande importance si le trafic n'est pas uniformément réparti. La **disponibilité** est la capacité à offrir le service sur de longues périodes et ceci sans interruption. Cela sous-entend qu'il faut mettre en place des redondances matérielles, logicielles, rendant le web service résistant aux pannes.

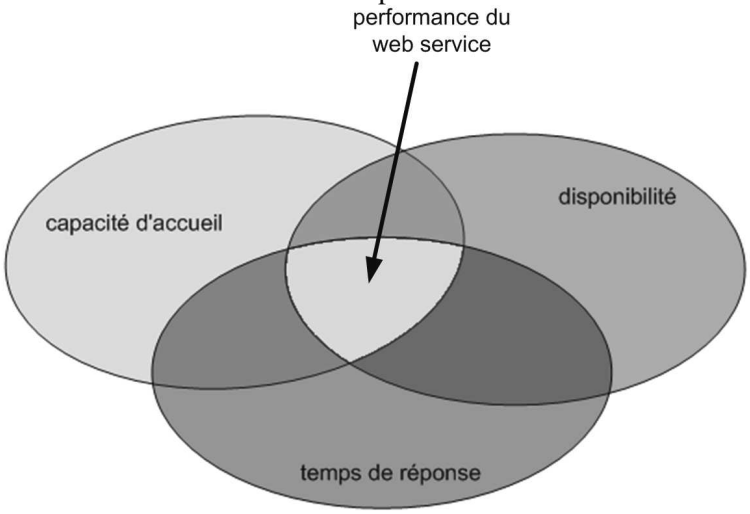

Figure 1-3. Performances du web service (point de vue de l'utilisateur)

  Le **temps de réponse** est simplement le temps nécessaire pour traiter la demande du client c'est-à-dire le temps qui s'écoule entre l'appel d'un web service et l'obtention d'un résultat fourni par le web service.

- le point de vue du concepteur qui conçoit et déploie un web service est différent. Celui-ci concentre son attention sur trois points complémentaires : **l'exploitabilité**, **l'évolutivité** et **l'extensibilité** qui, ensemble, définissent la qualité du web service (Figure 1-4).
  L'exploitabilité est la capacité à superviser facilement l'ensemble des services proposés et la capacité à appliquer des opérations de maintenance ou de changement de version avec des efforts raisonnables, et ceci, sans porter préjudice aux utilisateurs, c'est-à-dire sans dégrader la performance des web services du point de vue de l'utilisateur. La conception et le déploiement d'un web service sont deux opérations longues et difficiles. Une fois les web services déployés, les utilisateurs peuvent être nombreux, leurs besoins vont évoluer ; les techniques utilisées, d'un point de vue matériel et logiciel sur le serveur, devront suivre au fil des ans les nouvelles technologies. Le concepteur d'un web service doit donc se préoccuper de **l'évolutivité** de son système d'un point de vue fonctionnel et technique. **L'extensibilité** est aussi l'une de ses préoccupations majeures et cela dès la création. Il doit se pencher sur les problèmes éventuels liés à une augmentation importante du nombre d'utilisateurs et donc concevoir un système qui permet de moduler la capacité d'accueil à la hausse (on pense souvent à ce point en premier) mais aussi à la baisse.

L'utilisation d'un web service gratuit dans une application professionnelle pose le délicat problème des performances mais aussi de la gratuité dans le temps du web service choisi. On peut garder à l'esprit les web services fournis gratuitement par Google et qui sont devenus payants en 2012 (pour un nombre de requêtes par jour illimité).

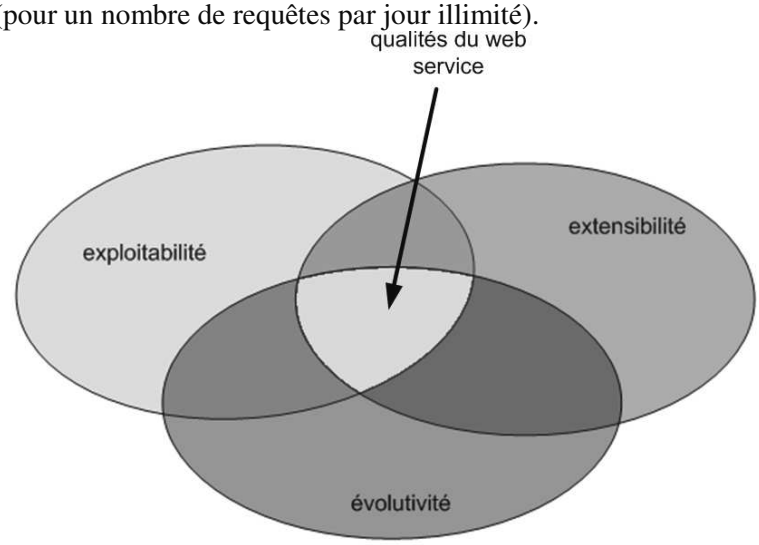

Figure 1-4. Qualités du web service (point de vue du concepteur)

De nombreux programmes informatiques ont été alors modifiés et le web service Google remplacé par un web service fourni, par exemple, par OpenStreetMap.
Il s'agit d'une réflexion à ne pas négliger faute de quoi, l'application initialement développée et parfaitement fonctionnelle peut devenir non fonctionnelle et engendrer de nouveaux coûts de développement.

1.3 Architecture générale des web services et des clients

La suite de ce chapitre sera organisée autour du schéma suivant décrivant tous les types de serveur et de client (Figure 1-5).

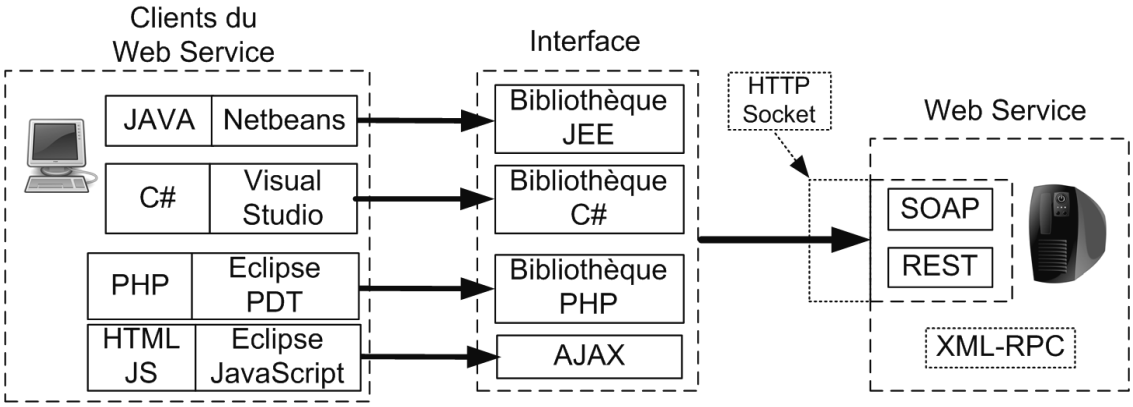

Figure 1-5. Schéma général des Web Services et des clients

La partie de droite décrit les trois types d'architecture d'un web service (XML-RPC, REST, et SOAP). La partie centrale, quant à elle, décrit les interfaces permettant d'utiliser les web Services (bibliothèques JEE, C#, PHP, AJAX). Enfin, dans la partie gauche, figurent tous les clients des web services avec les technologies utilisées (Java, C#, PHP ou encore JavaScript).

1.4 Les différentes architectures des web services

1.4.1 Introduction

Littéralement, utiliser un web service consiste à utiliser une méthode qui est disponible sur une machine distante accessible via Internet. Il existe de très nombreux moyens de fournir ce genre de service, mais généralement les web services existent sous trois architectures :

- l'architecture XML_RPC (historiquement la première) (Remote Procedure Call) ;
- l'architecture REST (REpresentational Stage Transfert);
- l'architecture SOAP (Simple Object Access Protocol).

Une mise en garde doit être adressée aux lecteurs concernant le sens des acronymes. Lorsque l'on parle **d'architecture SOAP** cela n'a pas réellement de sens car SOAP est un protocole de description des messages. Aussi, il vaut mieux comprendre cet acronyme comme une architecture client/serveur utilisant les technologies SOAP/HTTP/WSDL. La même remarque est aussi valable pour REST et XML-RPC. Le schéma, ci-dessous, représente les différentes architectures (Figure 1-6) qui seront exposées avec leurs particularités.

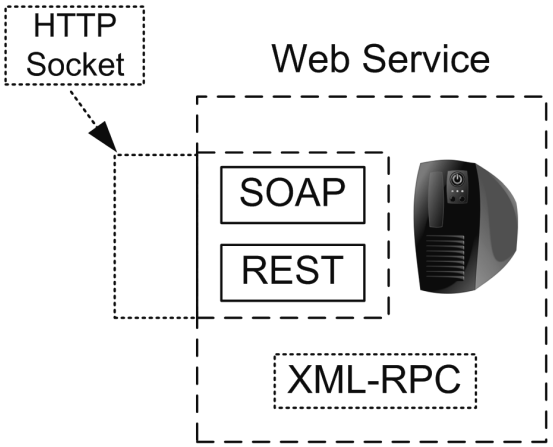

Figure 1-6. Les différentes architectures des Web Services

Parallèlement à ces trois architectures, il existe d'autres technologies qui permettent de faire communiquer un client et un serveur :

- DCOM spécifique à Microsoft ;
- RMI spécifique à Java ;
- CORBA créée par Sun, Oracle et IBM.

Par la suite, les trois architectures de web service XML_RPC, REST, SOAP vont être présentées. Ces présentations seront complétées par une introduction aux deux technologies utilisées par les web services à savoir le langage HTTP et le format de message XML.

1.4.2 L'architecture XML-RPC

1.4.2.1 Introduction

Cette architecture XML-RPC est apparue en 1998. Comme son nom l'indique, elle est composée de XML et de RPC. RPC (qui signifie en anglais Remote Procedure Call) est un protocole réseau. Son fonctionnement consiste, à partir d'un client, à faire appel à des procédures qui sont hébergées sur un serveur distant. Pour qu'une telle technologie fonctionne il faut :

- que le client possède une copie (nommée Stub) de l'objet distant ;
- que le client appelle les méthodes de la copie ;
- que le serveur possède un objet nommé Skeleton qui assure la connexion avec l'objet réellement appelé sur le serveur.

L'avantage principal de cette technologie est de permettre qu'un appel à une procédure distante soit identique à celui d'une procédure locale. Ainsi, pour le client toute la complexité de l'appel est cachée.

Pour qu'un appel RPC fonctionne il faut envoyer au serveur :

- le nom de la méthode à exécuter ;
- les paramètres de la méthode.

XML, quant à lui, est un langage informatique de balisage. Il est très utilisé pour l'envoi de message entre un serveur et un client sur internet et aussi par des applications de type client-serveur. Cette architecture est apparue lorsque l'idée d'appeler des services sur le web s'est fortement développée. Son fonctionnement est simple. La couche transport est assurée par le langage HTTP. En ce qui concerne l'envoi des requêtes et des réponses, c'est la méthode POST de HTTP qui gère les envois. Le langage de description des messages est le langage XML. Les lecteurs intéressés par cette architecture peuvent consulter le site suivant qui donne une spécification presque complète : http://xmlrpc.scripting.com/spec.html. Le schéma ci-dessous montre le fonctionnement de cette architecture XML-RPC (Figure 1-7). Dans un premier temps, un client envoie un document XML par la méthode POST du protocole HTTP. Le serveur le reçoit puis procède à un traitement. Lorsqu'il a terminé, il envoie à son tour la réponse sous la forme d'un document XML qui est transféré par le protocole HTTP.

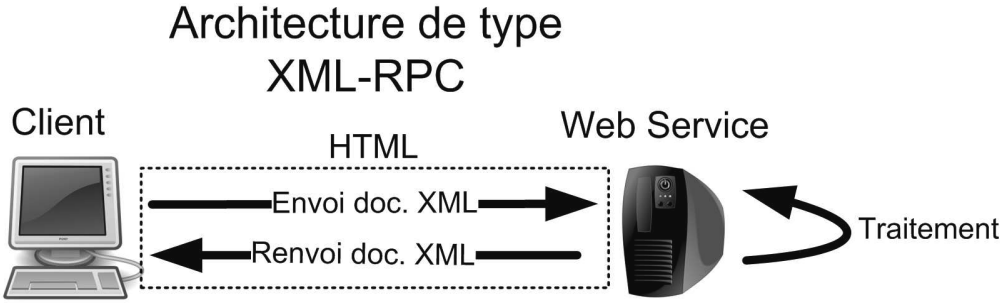

Figure 1-7. L'architecture XML-RPC

Afin qu'un appel RPC puisse avoir lieu, il faut fournir le nom de la méthode à appeler et les différents paramètres. Notons que de nombreux types sont supportés : les types simples (int, float…) et des types plus complexes (matrice, structures…).

Dans l'exemple ci-dessous, la méthode **getStateName** est appelée avec la valeur **41** comme paramètre.

```
POST /RPC2 HTTP/1.0
User-Agent: Frontier/5.1.2 (WinNT)
Host: betty.userland.com
Content-Type: text/xml
Content-length: 181
<?xml version="1.0"?>
<methodCall>
   <methodName>examples.getStateName</methodName>
   <params>
      <param>
         <value><i4>41</i4></value>
         </param>
      </params>
   </methodCall>
```

Un exemple de réponse possible est donné ci-dessous. Il s'agit d'un fichier XML ayant comme valeur de retour **terminated**. Le service (la méthode **getStateName**) a donc renvoyé comme réponse **terminated** à son exécution avec 41 comme paramètre d'appel.

```
<?xml version="1.0"?>
<methodResponse>
<params>
<param>
<value><string>terminated</string></value>
</param>
</params>
</methodResponse>
```

Le langage utilisé est simple mais un peu limitant car il ne prend pas en compte la sémantique du service appelé.

1.4.2.2 Exemple d'une architecture XML-RPC avec la méthode POST du protocole HTTP

L'exemple proposé sur le site http://xmlrpc-c.sourceforge.net fournit un exemple d'architecture XML-RPC en proposant un serveur XML-RPC qui renvoie le résultat d'une addition et d'une soustraction à partir de deux entiers. L'architecture XML-RPC utilise le protocole HTTP pour transmettre les messages entre le client et le serveur. L'exemple ci-dessous utilise la méthode POST pour transmettre le fichier XML contenant les paramètres de départ. La réponse se fait toujours par le protocole HTTP qui renvoie un fichier XML avec la réponse. Le schéma ci-dessous permet de mieux comprendre les échanges entre le client et le serveur (Figure 1-8).

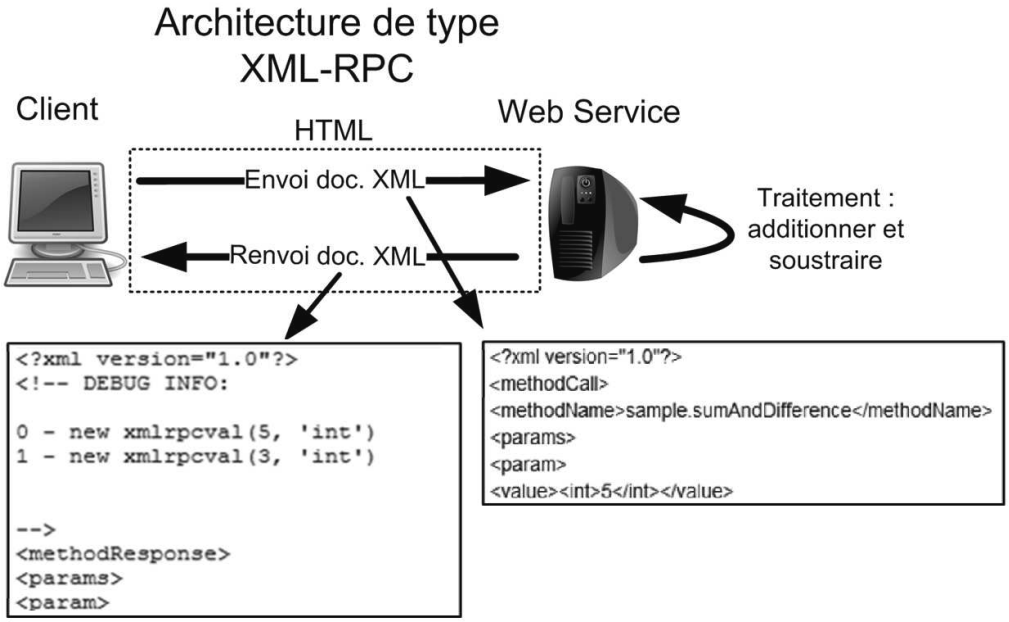

Figure 1-8. Exemple avec l'architecture XML-RPC

Pour reproduire l'exemple ci-dessus, il est recommandé d'utiliser l'extension de Firefox : Poster. En effet celle-ci offre de grandes facilités qui vont être illustrées par la suite.

Pour installer l'extension **Poster**, il faut utiliser le menu **Outils/Modules complémentaires** de Firefox, puis faire une recherche avec comme nom **Poster** afin d'obtenir l'extension à installer.

Figure 1-9. L'extension Poster

Après avoir double cliqué sur **Poster** pour l'ouvrir (en bas à droite de Firefox) il faut remplir le formulaire avec les informations du Tableau 1.1.

Tableau 1.1 : Tableau des paramètres pour appeler le Web Service xmlrpc-c

URL :	http://xmlrpc-c.sourceforge.net/api/sample.php
Content Type :	text/xml\r\n
Message :	<?xml version="1.0"?> <methodCall> <methodName>sample.sumAndDifference</methodName> <params> <param> <value><int>5</int></value> </param> <param> <value><int>3</int></value> </param> </params> </methodCall>

Dans cet exemple, les paramètres d'entrée sont les entiers 5 et 3 et on obtient la fenêtre de la Figure 1-10.

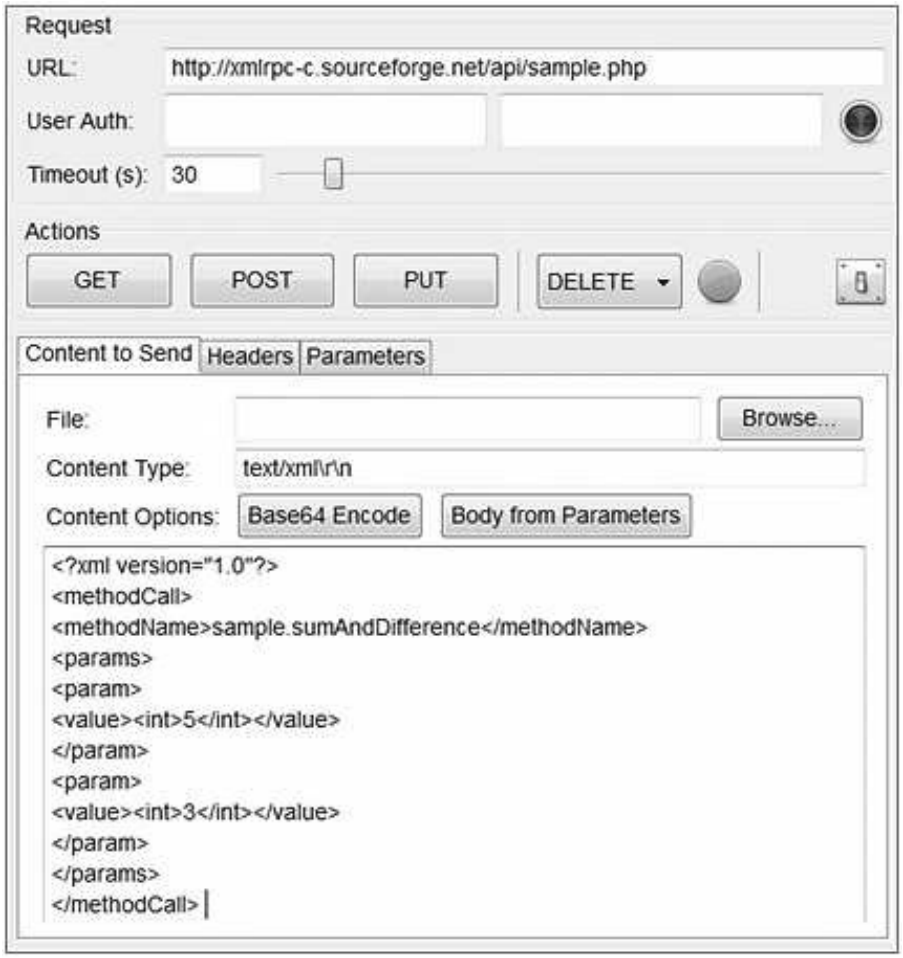

Figure 1-10. Saisie des données dans le plugin Poster

Ensuite, il faut cliquer sur le bouton **POST**. Ce clic provoque le transfert du fichier XML par la méthode POST de HTTP. On obtient, ensuite, une nouvelle fenêtre avec la réponse à la requête (Figure 1-11).