SI4- Polytech'Nice – Sophia
Jean-Yves Tigli, Diane Lingrand

2014-2015

Travaux Dirigés
Client / Serveur gSOAP en C/C++

1 Introduction

La boîte à outils gSOAP est destinée à de multiples plateformes de développement logiciel en C et C++ pour créer des serveurs et des clients de services Web. Les cibles peuvent être Windows en natif, Linux en natif, Mac-OS mais aussi pour des cibles plus légères comme le Raspberry Pi.

La boîte à outils gSOAP assure une faible empreinte mémoire. De nombreuses optimisations ont été appliquées pour réduire les besoins en ressources et pour accélérer l'analyse XML.

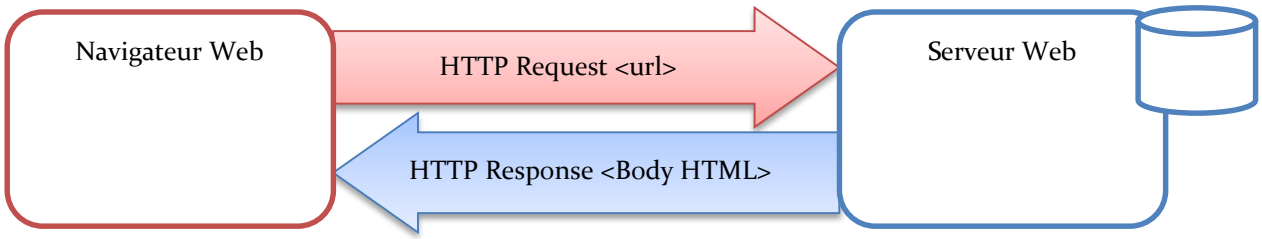
La boîte à outils gSOAP fournit un serveur Web HTTP(S) autonome (en plus d'un module Apache ou IIS).

Ce TD a donc pour objectif d'introduire les outils C/C++ gSOAP et leur mise en œuvre.

Pour commencer ce TD il vous faut récupérer les outils gSOAP sur SourceForge (http://sourceforge.net/projects/gsoap2), Vous devez télécharger la version stable 2.8.21 (http://sourceforge.net/projects/gsoap2/files/gSOAP/) et l'installer sur votre machine sous Windows.

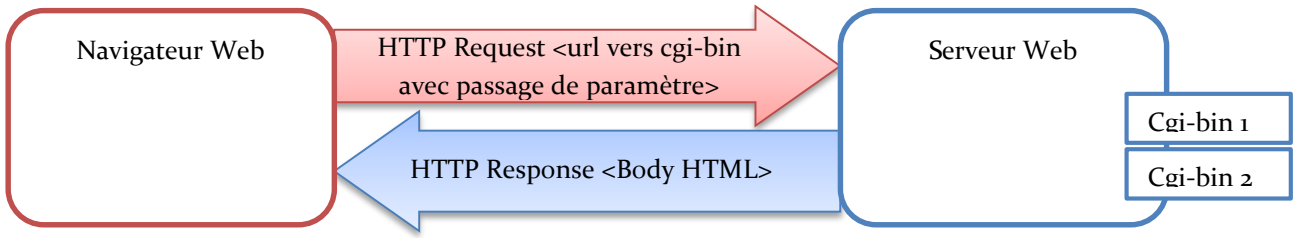
Vous trouverez tous les détails sur les outils et librairies de gSOAP dans le « gSOAP 2.8.21 User Guide » sur le lien http://www.cs.fsu.edu/~engelen/soapdoc2.html.

Dans les précédents TD nous avons vu comment nous pouvions récupérer dans un navigateur Web des données au format HTML stockées dans le système de fichier du serveur Web à partir d'un URL.



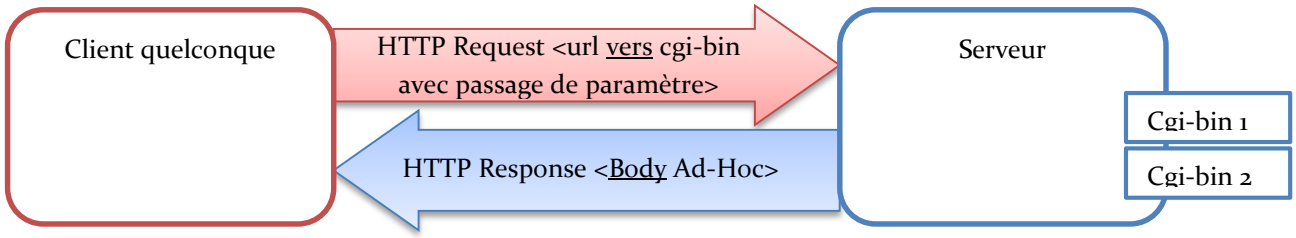
Ensuite nous avons vu comment générer des pages Web dynamiques à partir de fichiers exécutés et générant des contenus HTML à la volée : les cgi-bin. Ces pages HTML sont alors visualisable depuis le client utilisé : un navigateur.

Dans le cas où nous ne limitons plus les clients d'un serveur Web à des navigateurs, le contenu des données échangées entre client et serveur Web/cgi-bin peuvent prendre alors n'importe quel format (Ad-Hoc). Nous pouvons



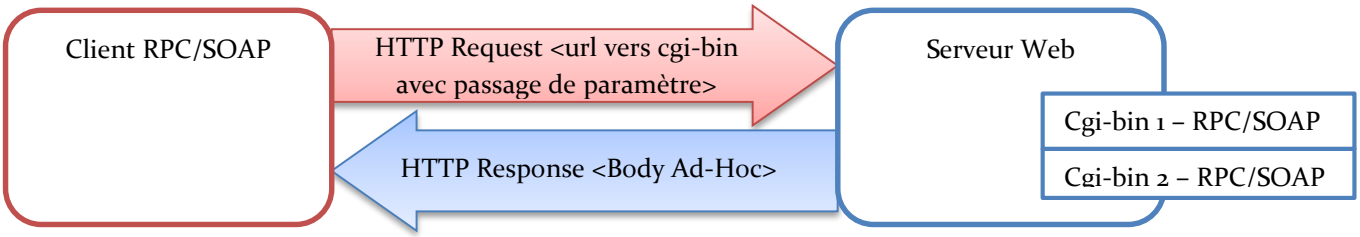
Contributeurs par ordre alphabétique : Nicolas Ferry, Tristan Glatard, Jean-Yves Tigli

alors utiliser cette approche pour implémenter de communications type RPC (Remote Procedure Call) entre applications réparties type Client / Serveur.



Dans le cas des Service Web, un certain nombre de standard spécifient le protocole RPC mis en œuvre et le format d'échange des données correspondant.

Nous verrons plus en détail les principes du format RPC/SOAP standardisé par le W3C (http://www.w3.org/TR/soap/) en cours.



En attendant, dans ce TD et à l'aide des libraires et des outils gSOAP, nous allons mettre en place ce protocole RPC/SOAP de communication entre un client et un serveur Web/cgi-bin.

2 Création d'un Service Web

Cette section a pour objectif d'introduire les outils C/C++ gsoap pour créer un service web et un client pour y accéder en C.

Pour utiliser les outils gSOAP, il vous faudra ensuite, sous un éditeur de commandes en ligne (ex. cmd.exe ou un terminal sous cygwin), rajouter (%PATH) le chemin vers les exécutable de gSOAP. Vous pouvez le faire dans Panneau de Configuration/Système/Paramètres Système avancés/ Variables d'environnement. Ainsi %PATH% sera déjà positionné au lancement de tout « cmd » suivant.

Nous allons mettre en place un service Web qui ne possède qu'une méthode de nom « ns_display » dont le but est uniquement d'afficher coté serveur un texte un certain nombre de fois (int nb).

Editer dans un nouveau répertoire SOAP2, le fichier service.h suivant:

```
//gsoap ns service name: messg
//gsoap ns service style: rpc
//gsoap ns service encoding: literal
//gsoap ns service location: http://localhost:8060
//gsoap ns schema namespace: urn:messg
```

```
int ns__display( int nb, char *message, struct ns__noResponse {} *out );
```

gSOAP utilise des directives au début du fichier d'inclusion afin de récupérer certaines informations comme le nom du service (messg), le type du service (rpc), la localisation (http://localhost:8060) ou encore l'espace de nommage utilisé (messg).

La syntax de la fonction ns_display est la suivante :

- Le code retour int de la fonction correspond au status retourné par l'invocation à distance (exemple « SOAP_OK »)
- ns_display : nom de la fonction.
- int nb, et char* message : les paramètres de la fonction invoquées à distance.
- struct ns_noResponse {} *out : un pointeur sur les données retournées par la fonction invoquée à distance (le type dépend de leur format).Ici il n'y a pas de donnée retournée.

Utiliser la commande soapcpp2 pour générer l'ensemble des fichiers. N'oubliez pas les options -c pour générer du code source exclusivement en langage C, et –p afin de préciser le préfixe des fichiers générés.

Un ensemble de fichiers sont alors générés dont :

messgC.c	Cœur de l'encodage/décodage des trames SOAP
messgServer.c	Partie serveur du service web
messgClient.c	Partie cliente du service web
messgServerLib.c	Pour générer une librairie avec un serveur SOAP
messgClientLib.c	Pour générer une librairie avec un client SOAP
messgH.h	Principal fichier d'inclusion du service
Messg.nsmap	Détail de tous les espaces de nommage utilisés dans le service
messg.wsdl	Description du service web dans le langage descritpif WSDL

3 Application serveur

La mise en place d'un serveur SOAP répond à la logique suivante (vous trouverez le détail de ces fonctions dans le document soapdoc2.pdf du répertoire gsoap\doc) :

- Initialisation du contexte SOAP avec *soap_init* (initialisation du runtime)
- Assignation d'un espace de nommage particulier avec *soap_set_namespaces* (nous avons utilisé l'option –n précédemment)
- Création de la socket mère de connexion sur le port 8060 avec *soap_bind*
- Dans une boucle de connexion :
  - On accepte une connexion cliente avec *soap_accept* (dans cette exemple, on rend l'attente non bloquante) ;
  - On traite la requête avec *messg_serve* (nous sommes dans l'espace de nommage « messg »)

- On désalloue les données temporaires mémoire réservées pour le traitement de la requête avec *soap_destroy*
- Fermeture de la connexion socket avec *soap_end*
- On ferme les connexions sockets clientes et serveur et on enlève les fonctions callback avec *soap_done*.

Créez maintenant un fichier serveur.c pour la réception et le traitement des appels ainsi que l'implémentation du corps de la fonction distante ns__display(...) sur le modèle ci-dessous. Complétez le code et en particulier le corps de la fonction ns__display(...).

```
#include<stdio.h>
#include<stdlib.h>


// serveur SOAP
#include "messgH.h"
#include "messg.nsmap"

// permet de supporter plusieurs clients et plusieurs serveurs
extern SOAP_NMAC struct Namespace namespaces[];

int ns__display(
        struct soap *p_soap,          // contexte d'exécution du service web
        int p_nb,                   // entier (nb)
        char *p_chaine,               // chaine de caractere (message)
        struct ns__noResponse *p_out  // pas de réponse (out)
) {

// A compléter
  }

int main(
      int p_argc,
      char *p_argv[]
) {
        struct soap v_soap;     // contexte du service SOAP

       // on initialise la socket
// A compléter
      // on assigne l'espace de nommage
// A compléter

        // on crée la socket mère de connexion

        if (soap_bind(&v_soap,NULL,8060,100) < 0) { return -1; }

        // on gère une boucle infinie pour recevoir les requêtes
        for (;;) {
                v_soap.accept_timeout = 1; // on rend l'attente non bloquante
                if (soap_accept(&v_soap) < 0) { continue; }
                soap_serve(  &v_soap );
                soap_destroy( &v_soap );  // dealloc data
                soap_end(     &v_soap );  // dealloc data and cleanup
```

```
            }
        soap_done( &v_soap );    // détache la strcut soap
    }
```

Sous Visual Studio, créez un projet Visual C++ « vide ». Inclure dans ce projet le fichier serveur.c complété et l'ensemble des fichiers nécessaires à sa compilation (fichiers générés par gsoap spécifiques au service web considéré et les deux fichiers stdsoap2.c et stdsoap2.h de la bibliothèque gsoap/C++).

Soit : messg.nsmap, messgC.c, messgH.h, messgServer.c, messgStub.h, **stdsoap2.c, stdsoap2.h**, serveur.c
Attention vous aurez peut-être à gérer d'autres dépendances, par exemple avec la libwsock32.a.

Après compilation, exécutez votre projet serveur.exe et vérifier la présence du serveur Web sur http://localhost:8060.
Quel est le contenu du document XML qui vous est retourné ? De quelle erreur s'agit-il et pourquoi ?

4 Application cliente

Passons maintenant à la réalisation de notre client.

Créez et complétez le fichier suivant client.c :

```
#include "messgH.h"
#include "messg.nsmap"

extern SOAP_NMAC struct Namespace namespaces[];

int main(int p_argc,char *argv[]
) {

  struct ns__noResponse out;
  struct soap v_soap;

  soap_init( &v_soap );
  soap_set_namespaces( &v_soap, messg_namespaces );

// Appel de la méthode ns__display distante avec la fonction
// int soap_call_ns__display(struct soap *soap, const char *soap_endpoint, const char
// *soap_action, int nb, char *message, struct ns__noResponse *out);

  // A compléter

  if (v_soap.error) { soap_print_fault(&v_soap, stderr); }
  soap_end( &v_soap );
  return(0);
  }
```

Remarquons que l'appel à la fonction distante se fait au travers la fonction préfixée par soap_call_.... .
Le traitement des erreurs se fait au travers la variable v_soap.error et la fonction soap_print_fault.

Compilez le projet et exécutez *client.exe 3 « Bonjour »* . Que remarquez vous sur le serveur ?

Travaux Dirigés
Client / Serveur gSOAP en C/C++

5 Génération statique du Proxy avec gSOAP

Comme nous le verrons en cours, il est possible de décrire un service Web à partir d'un fichier XML en lieu et place du .h utilisé ci dessus.

En fait cela permet d'avoir un fichier de description de l'interface du service indépendante du langage d'implémentation. En effet le fichier « .h » est dédié à la description des entêtes de fonctions en C.

Il existe donc un format dérivé d'XML appelé WSDL (Web Service Description Language) que nous pouvons utiliser pour générer le code client et serveur d'un service Web.

Il faut donc commencer par récupérer le fichier de description WSDL du service web que nous allons utiliser pour ******les exercices suivants. Nous prendrons un service existant déjà exposé sur le Web.

Il s'agit d'un service web gratuit de WebserviceX.NET : ConversionRate dont vous pouvez charger le fichier wsdl à l'url : http://www.webservicex.net/currencyconvertor.asmx?WSDL. Sauvegardez-le dans un fichier CurrencyConvertor.wsdl et dans un répertoire SOAP.

Générez à partir du WSDL du web service dans le répertoire SOAP du projet, le header WSCurrencyConv.h correspondant, grâce à la commande :

```
wsdl2h.exe -o WSCurrencyConv.h CurrencyConvertor.wsdl
```

Ce fichier n'est qu'un fichier intermédiaire qui sera utilisé pour générer tous les fichiers nécessaires à la classe proxy du service web.

Générez ensuite les fichiers nécessaires à la classe CurrencyConvertorSoap (Cf. pour un client, le fichier soapCurrencyConvertorSoapProxy.h et pour un serveur, le fichier soapCurrencyConvertorSoapObjet.h), grâce à la commande :

```
soapcpp2 -I"……\gsoap\import" WSCurrencyConv.h
```

6 REFERENCES BIBLIOGRAPHIQUES

Robert van Engelen. 2004. Code generation techniques for developing light-weight XML Web services for embedded devices. In *Proceedings of the 2004 ACM symposium on Applied computing* (SAC '04). ACM, New York, NY, USA, 854-861. DOI=10.1145/967900.968075 http://doi.acm.org/10.1145/967900.968075