[laboratori] Lab 04: Serveis Web amb SOAP + WSDL

Lab 04: Serveis Web amb SOAP + WSDL

Introducció

En aquesta quarta sessió de laboratori ens introduirem a la implementació i utilització de Serveis web basats en SOAP+WSDL. De fet, el protocol SOAP no l'utilitzarem directament, ja que usarem una llibreria que farà les crides de baix nivell (SOAP sobre HTTP) per nosaltres. Més concretament, per un constat tindrem un script PHP -executant-se en un servidor Apache- que implementarà un servei web definit en WSDL. Per l'altre, hi haurà un client, també implementat en PHP, que cridarà les operacions de servei web.

Tasques a Realitzar

Prèvia

Se suposa que ja heu entrat als PCs escollint LINUX com a Sistema Operatiu. Les tasques s'han de dur a terme mantenint l'ordre seqüencial amb què són descrites a continuació.

Tasca #1 (10 punts si es completa abans de la fi de la sessió de laboratori)

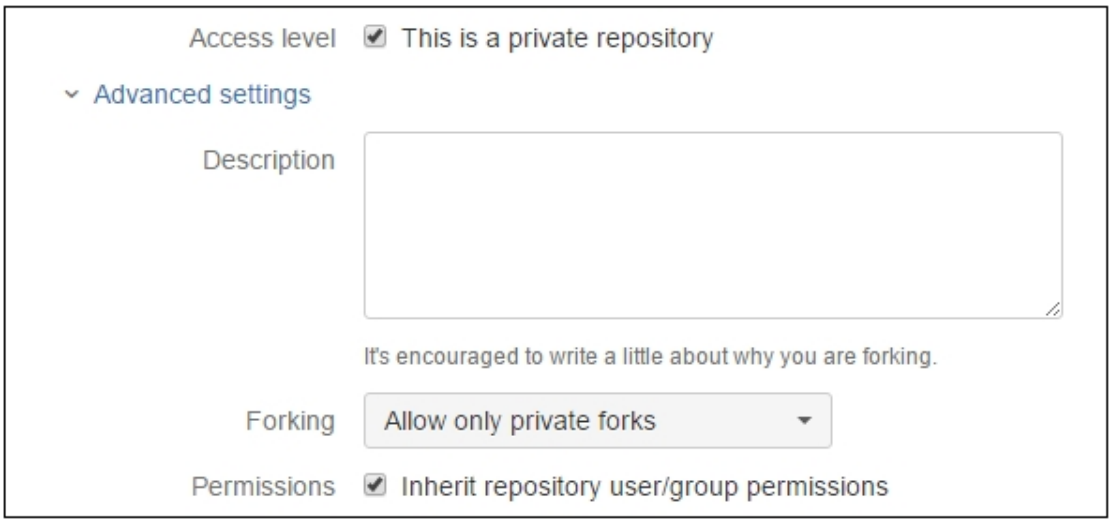
Comprovació prèvia

Se suposa ja teniu el servidor apache instal·lat. Si no és així, aneu al primer apartat de la primera tasca de la segona sessió de laboratori.

Descàrrega i execució del projecte "waslab04"

1. Aneu al repo https://bitbucket.org/fib_was/waslab04 i feu-ne un "Fork". Un dels dos membres de la parella s'haurà de "loguejar" amb les seves credencials de bitbucket.org. No cal que canvieu el nom del vostre

repositori però sí que es necessari que les opcions "**This is a private repository**" (Access Level) i "**Inherit repository user/group permissions**" (Permissions) **estiguin seleccionades** tal com es mostra a la figura:



2. Cliqueu sobre el botó "Clone" del vostre repositori (no l'original!) i copieu-ne la comanda git.
3. Des del terminal de Linux, aneu al vostre directori dades/apache2/htdocs i executeu la comanda git que heu copiat abans (p.ex: git clone https://el_vostre_username@bitbucket.org/el_vostre_username/waslab04.git)
4. Arranqueu apache2: $HOME/dades/apache2/scripts/apache2.init start (des del terminal de Linux)
5. Feu php waslab04/client_04.php des del terminal o, si ho preferiu, http://localhost:8080/waslab04/client_04.php des del navegador. En qualsevol cas, hauríeu d'obtenir un text pla semblant a:

```
63.5 Fahrenheit ==> 17.5 Celsius [Server TimeStamp: Monday 14th of March 2016 @ 12:31:56 PM]
```

Primer commit

1. Editeu el fitxer waslab04/README.md, poseu-hi els vostres noms i cognoms i salveu-lo.

2. Des del terminal de Linux feu:

```
$ cd waslab04
$ git add README.md
$ git commit -m "tasca #1 finalitzada"
$ git push origin
```

Tasca #2 (15 punts si es completa abans de la fi de la sessió de laboratori)

Si doneu un cop d'ull al codi de client_04.php i server_04.php, sembla que la cosa és prou senzilla: client_04.php invoca una operació, FahrenheitToCelsius, i server_04.php rep la invocació, l'executa i retorna el resultat a client_04.php. Aparentment, no hi ha cap restre ni de SOAP ni d'HTTP. Dos objectes, $sClient a client_04.php i $server a server_04.php, s'encarreguen d'amagar-nos els detalls. En l'argot RPC, aquests objectes són l' *stub* i l'*skeleton*, respectivament.

Ara bé, en aquesta tasca volem veure els detalls que queden amagats. Més concretament, volem veure la petició HTTP que $sClient envia a server_04.php i la resposta HTTP que d'aquest rep. Tant la petició com la resposta inclouen, en el seu cos, el missatges SOAP corresponents. Per visualitzar aquesta informació caldrà que modifiqueu l'script client_04.php allà on indiquen els comentaris. Necessitareu invocar quatre operacions sobre $sClient: __getLastRequestHeaders (retorna la capçalera de la petició HTTP), __getLastRequest (retorna el cos de la petició HTTP, és a dir, la petició SOAP en sí), __getLastResponseHeaders (retorna la capçalera de la resposta HTTP) i __getLastResponse (retorna el cos de la resposta HTTP, és a dir, la resposta SOAP en sí). En el cas de les capçaleres, només cal fer un echo del resultat retornat per les corresponents operacions. Pel que fa al contingut SOAP (XML) retornat per les altres dues operacions, utilitzeu la funció xmlpp, ja implementada, per mostrar-ne el contingut d'una manera més "amigable". Al final, hauríeu de ser capaços d'obtenir quelcom semblant a capture01.txt.

Un cop ja funciona, des del terminal de Linux feu:

```
$ git add client_04.php
$ git commit -m "tasca #2 finalitzada"
```

$ git push origin

Tasca #3 (15 punts si es completa abans de la fi de la sessió de laboratori)

A part de l'operació FahrenheitToCelsius, server_04.php té implementada una altra operació, CurrencyConverter. A més, client_04.php ja està quasi preparat per invocar-la, ja que només falta "descomentar" el codi corresponent. Tot està quasi a punt perquè client_04.php utilitzi aquesta operació, només manca especificar-la al fitxer WSLabService.wsdl. Anem per passos:

WSLabService.wsdl:

Caldrà ampliar les definicions d'aquest fitxer per tal d'afegir l'especificació de l'operació CurrencyConverter. Més concretement:

- S'han d'afegir dos nous "message", un per fer la petició i un altre per retornar la resposta:
  - El primer missatge tindrà tres paràmetres (from_Currency, to_Currency i amount) que seran xsd:string els dos primers i xsd:double el darrer
  - El segon missatge tindrà un únic paràmetre, de tipus xsd:double
- Al porType "WSLab_PorType" s'ha de definir la nova operació, CurrencyConverter, amb els dos missatges input i output corresponents.
- Al binding "WSLab_Binding" hi afegireu també l'operació CurrencyConverter. En aquest cas, només cal fer copy&paste de l'operació que ja hi és i substituir les dos ocurrències de FahrenheitToCelsius per CurrencyConverter.

server_04.php:

"Descomenteu" $server->addFunction("CurrencyConverter");

client_04.php:

"Descomenteu" les línies corresponents a la Tasca #3. A l'executar-lo, us hauria de sortir una nova línia a la sortida generada per client_04.php (el valor concret de la conversió pot variar segons les fluctuacions en les cotitzacions de les divises):

```
100 EUR ==> 722.3 CNY
```

Per acabar, des del terminal de Linux feu:

```
$ git add WSLabService.wsdl *.php
$ git commit -m "tasca #3 finalitzada"
$ git push origin
```

Tasca #4 (25 punts si es completa abans de la fi de la sessió de laboratori)

En aquesta tasca jugarem una mica amb els tipus complexos de WSDL. Per fer-ho, haureu d'especificar (WSLabService.wsdl), implementar (server_04.php) i invocar (client_04.php) una nova operació anomenada CurrencyConverterPlus. Aquesta operació és una variant de la que heu vist a la tasca anterior. La diferència està en què en comptes de passar-li una sola "curreny" de destí, ara se li passarà una llista de "currencies". El resultat, per tant, també serà una llista que contindrà per cada "currency" de destí el valor de conversió corresponent. Els detalls, a continuació.

WSLabService.wsdl:

Com en la tasca anterior, caldrà definir dos nous "message" i afegir la nova operació al porType i al binding. Però com que els tipus dels paràmetres dels dos missatges seran tipus no standard, caldrà definir primer aquests nous tipus complexos. Anem a pams, doncs.

- A la part marcada amb el comentari <!-- ComplexType Definitions --> definirem els "ComplexType" que necessitarem. En concret són quatre:
  - Un complexType que podeu anomenar "ArrayOfString". En aquest cas el podeu copiar directament d'aquest exemple: wsdl.html#ex1.
  - Un complexType, que podeu anomenar "ConversionRequest", per exemple, que serà una "sequence" de tres elements: "from_Currency",

"to_Currencies" i "amount". El primer element serà de tipus "xsd:string", el segon "xsd1:ArrayOfString" i el tercer "xsd:double". Mireu aquest exemple per veure com es fa: wsdl.html#ex4.

- Un complexType que podeu anomenar "ConversionResult", que també serà una "sequence" com l'anterior. Però en aquest cas, tindrà només dos elements, "currency" i "amount", que seran "xsd:string" i "xsd:double" respectivement.
- Un complexType que podeu anomenar "ArrayOfConversionResult", que es defineix de manera similar a "ArrayOfString", però substituint "xsd:string" per "xsd1:ConversionResult" on calgui.

- S'han d'afegir dos nous "message", un per fer la petició i un altre per retornar la resposta:
  - El primer missatge tindrà un únic paràmetre de tipus "xsd1:ConversionRequest".
  - El segon missatge tindrà un únic paràmetre de tipus "xsd1:ArrayOfConversionResult".
- Al porType "WSLab_PorType" s'ha de definir la nova operació, CurrencyConverterPlus, amb els dos missatges input i output corresponents.
- Al binding "WSLab_Binding" hi afegireu també l'operació CurrencyConverterPlus de la mateixa manera com s'ha fet a la tasca anterior.

server_04.php:

Allà on marca el comentari corresponent heu d'afegir el codi per implementar la funció CurrencyConverterPlus. Es tracta de generar, per cada currency de destí que hi ha al paràmetre d'entrada, un objecte amb dos camps, currency i amount, i que es correspon amb al complexType "ConversionResult". El valor del camp currency és el mateix que el de la

currency de destí, i el valor amount el podeu obtenir cridant la funció CurrencyConverter amb els paràmetres corresponents. Així, doncs, el resultat a retornar per l'operació CurrencyConverterPlus serà una llista (array) d'aquests objectes. Mireu els exemples 2 i 5 del document wsdl.html per veure com s'utilitzen aquestes estructures de dades amb PHP. De totes maners, per recòrrer un array és més fàcil utilitzar el "foreach".

A més, caldrà afegir també aquesta funció a l'objecter $server (addFunction).

client_04.php:

Allà on està marcat pels comentaris, feu la crida a l'operació CurrencyConverterPlus. Fixeu-vos que el paràmetre que li cal passar és de tipus complex (ConversionRequest). Per tant, tal com heu fet abans, haureu de crear un objecte amb els tres camps corresponents. Trieu vosaltres mateixos el valor d'aquest camps. Trobareu una llista de currencies vàlides a http://currencies.apps.grandtrunk.net/. No feu l'array massa llarg, ja que pot trigar una estona. El resultat que retorna l'operació s'ha de mostrar per pantalla, com, per exemple:

```
1000 CNY
        ==> 138.45 EUR
        ==> 204.03 CAD
        ==> 443.71 TRY
```

Per acabar, des del terminal de Linux feu:

```
$ git add WSLabService.wsdl *.php
$ git commit -m "tasca #4 finalitzada"
$ git push origin
```

Tasca #5 (Fins a 35 punts si es completa com a molt tard a les 22:00 del dimarts 29.03.2016)

Finalment, en aquesta tasca finalitzareu la implementació de la web app "WASLab04 Weather Mashup" (http://localhost:8080/waslab04/weather.html), que ha de permetre consultar la informació meteorològica de diferents ciutats del món a partir de les operacions que trobareu a http://www.webservicex.net/globalweather.asmx. Aquesta web app que heu d'acabar consisteix en un pàgina HTML, weather.html, que en carregar-se mostrarà quelcom semblant a snapshot_5_1.png. Com podeu veure, a la part esquerra hi apareix un llistat de ciutats d'un determinat país. La primera vegada que es carrega la pàgina, es mostra un país per defecte, però aquest es pot canviar mitjançat el selector. Si es clica a sobre d'una de les ciutats, s'ha de mostrar la informació meteorològica d'aquella ciutat a la part dreta, tal com es mostra a la snapshot_5_2.png. Més concretament, haureu de completar la implementació de les funcions JavaScript showCities i showWeather del fitxer weather.js i la de les pàgines PHP city_names.php i city_info.php. Ja us aviso que us caldrà repassar també alguns dels laboratoris anteriors. Anem a pams.

Funció **showCities** de weather.js:

> Aquesta funció es crida quan els carrega la pàgina weather.html i cada cop que es canvia el país del selector. Nota: això ja està implementat.
>
> Aquesta funció haurà de fer una petició HTTP a la pàgina PHP city_names.php passant-li el nom del país, utilitzant l'objecte XMLHttpRequest.
>
> city_names.php retornarà un array d'strings en format JSON i llavors aquests strings (noms de ciutat) s'hauran de mostrar a la part esquerra de la pàgina de tal manera que quan es cliqui sobre un d'ells es cridi la funció showWeather.

Funció **showWeather** de weather.js:

Aquesta funció es cridarà quan es cliqui a sobre d'una de les ciutats que apareixaran a la part esquerra de la pàgina weather.html.

Aquesta funció haurà de fer una petició HTTP a la pàgina PHP city_info.php passant-li els noms de la ciutat i del país, utilitzant l'objecte XMLHttpRequest.

city_info.php retornarà un objecte JavaScript en format JSON i llavors s'haurà de mostrar tot el seu contingut a la part dreta de la pàgina weather.html.

city_names.php:

Aquesta pàgina php processarà la petició HTTP de la funció showCities, la qual li passarà el nom del país com a paràmetre.

Aquesta pàgina cridarà l'operació **GetCitiesByCountry** tenint en compte tal com està definida a http://www.webservicex.net/globalweather.asmx?WSDL. Aneu em compte, sobre tot, amb les estructures de dades necèssaries per enviar i rebre la informació d'aquesta operació (tipus complexos).

Aquesta pàgina retornarà la informació obtinguda de GetCitiesByCountry en format JSON (array d'strings). Exemple: city_names.txt.

city_info.php:

Aquesta pàgina php processarà la petició HTTP de la funció showWeahter, la qual li passarà els noms de la ciutat i del país com a paràmetres.

Aquesta pàgina cridarà l'operació **GetWeather** tenint en compte tal com està definida http://www.webservicex.net/globalweather.asmx?WSDL. Aneu em compte, sobre tot, amb les estructures de dades necèssaries per enviar i rebre la informació d'aquesta operació (tipus complexos).

Aquesta pàgina retornarà la informació obtinguda de GetWeather en format JSON (objecte). Exemple: city_info.txt.

Commits & pushes: Feu-ne de manera freqüent.

Lliurament de la Pràctica

Si heu creat el vostre repositori de bitbucket.org tal com s'indicava a la Tasca #1, jo també hi tinc accés. Per tant, el repositori en sí és el vostre "lliurament". Això sí, **aneu comprovant de tant en quant que els vostres commits es propaguen correctament al vostre repositori de bitbucket.org**.

Avaluació

Al principi de cada tasca ja s'especifica quants punts val si es realitza en el termini establert. En el cas de les tasques #1-4, si no s'acaben dins de la sessió de laboratori, es poden finalitzar amb posterioritat però, en aquest cas, **valdran 5 punts** cadascuna.