Ejercicios. Invocación de Servicios Web SOAP

Índice

1. Repositorio Mercurial para los ejercicios

Todos los proyectos para los ejercicios de este módulo los crearemos, por ejemplo, en el directorio NetBeansProjects/servc-web-expertojava.

En Bitbucket debemos crear en nuestra cuenta el repositorio **servc-web-expertojava**. En Netbeans lo configuraremos como repositorio *push* por defecto, pulsando el botón derecho sobre cualquier proyecto y seleccionando la opción *Mercurial > Properties >... e introduciendo sus datos (tal y como ya habéis hecho para los ejercicios del módulo de componentes enterprise).*

Cuando inicializamos un repositorio de Mercurial local desde Netbeans, éste crea el fichero .hginore. Dado que los dos primeros ejercicios los vamos a hacer sin utilizar Netbeans, podemos inicializar el repositorio de Mercurial local creando un proyecto maven java en la carpeta servc-web-experto-java. Después seleccionamos dicho proyecto e inicializamos el repositorio local con la opción *Team > Mercurial > Initialize Project* indicando el directorio NetBeansProjects/servc-web-expertojava.

Con esto se habrá creado el fichero `.hginore` en el directorio *NetBeansProjects/servc-web-expertojava*. Tal y como vimos en los ejercicios de la primera sesión del módulo de componentes enterprise, tendremos que ejecutar los comandos `hg add .hgignore` y `hg commit -m "Añadido .hgignore"`, desde el directorio NetBeansProjects/servc-web-expertojava para incluir añadir dicho fichero en el repositorio de Bitbucket.

Para subir los proyectos de los ejercicios que no requieren Netbeans a Bitbucket, podemos abrirlos desde Netbeans y utilizar las opciones `Mercurial > Commit` (desde el menú contextual de los proyectos), y posteriormente haremos `Mercurial > Share > Push to default .`

2. Clientes para servicio web hola

En las plantillas, disponéis de un war con la implementación del servicio web *hola*. Despliega dicho fichero: *HolaMundo.war* en el servidor de aplicaciones Glassfish para poder realizar el ejercicio. Puedes hacerlo desde línea de comandos, primero arranca el servidor con `/opt/glassfish-3.1.2.2/bin/asadmin start-domain domain1`, y después despliega el fichero con `/opt/glassfish-3.1.2.2/bin/asadmin/deploy HolaMundo.war`. En la consola de administración de glassfish (http://localhost:4848/) verás que se ha añadido la aplicación *HolaMundo* a la lista de aplicaciones ya desplegadas (también podrías comprobarlo desde línea de comandos con `/opt/glassfish-3.1.2.2/bin/asadmin list-applications`). En el nodo de Aplicaciones verás el componente *HolaMundo*, pincha en *Ver punto final*. Aparecerá la dirección en dónde se ha generado el fichero wsdl asociado al servicio (dirección

/HolaMundo/hola?wsdl). También puedes ver que glassfish ha generado un *driver* de pruebas en */HolaMundo/hola?Tester*. Ejecuta el test para probar el método que proporciona el servicio.

Implementa un cliente, con nombre *HolaMundoJavaClient*, desde una clase Java (utiliza Maven desde línea de comandos) para el servicio hola a partir de su descripción wsdl (ver apartado *Invocación de servicios web JAX-WS desde una clase Java con Maven*). (0,5 puntos)

Cuando crees el arquetipo de Maven, utiliza los siguientes nombres para:

- groupId: expertoJava
- artifactId: HolaMundoJavaClient

Implementa un cliente, desde un *jsp* (utiliza Maven desde línea de comandos) para el servicio hola a partir de su descripción wsdl (ver apartado *Invocación de servicios web JAX-WS desde una aplicación Web con Maven*, subapartado *Invocación del servicio web desde una página JSP*). (0,5 puntos)

Cuando crees el arquetipo de Maven, utiliza los siguientes nombres para:

- groupId: expertoJava
- artifactId: HolaMundoWebClient

3. Cliente para el servicio web Calculadora

En las plantillas, disponéis de un war con la implementación del servicio web *Calculadora*. Despliega dicho fichero: *Calculadora.war* en el servidor de aplicaciones Glassfish para poder realizar el ejercicio. Accediendo a la consola de administraciónd de Glassfish verás que se ha añadido la aplicación *Calculadora* a la lista de aplicaciones ya desplegadas. Verás el componente *Calculadora* de tipo *Servlet*, pincha en *Ver punto final*. Aparecerá la dirección en dónde se ha generado el fichero wsdl asociado al servicio (dirección */Calculadora/calculadora?wsdl*). Fíjate que, en este caso, en el wsdl se declara el mensaje <fault message> con el nombre *NegativeNumberException*. También puedes ver que Glassfish ha generado un *driver* de pruebas en */Calculadora/calculadora?Tester*. Ejecuta el test para probar el método que proporciona el servicio.

Nota

El cliente de prueba no trata correctamente la recepción de un SOAP Fault como respuesta, ya que la excepción que genera éste hace que falle la misma aplicación de pruebas. Para poder tratar la excepción de forma correcta deberemos hacerlo con un cliente propio.

Implementa un cliente, desde un *servlet* utilizando Maven para el servicio Calculadora a partir de su descripción wsdl(ver apartado *Invocación de servicios web JAX-WS desde una aplicación Web con Maven*). (1 punto)

Cuando crees el arquetipo de Maven, utiliza los siguientes nombres para:

- groupId: expertoJava
- artifactId: CalculadoraWebClient

El nombre del componente para el plugin de Glassfish será: *CalculadoraClient*. Configurar la raiz del contexto como *CalculadoraWeb* en el fichero *glassfish-web.xml*.

La url del servlet indicada en el atributo *url-patterns* de la anotación *@WebServlet* será: *calculadoraWebCliente*.

A continuación mostramos ejemplos de llamadas al servicio y el resultado por pantalla correspondiente:

Caso 1. Introducimos los parámetros en la llamada http:

```
http://localhost:8080/CalculadoraWeb/calculadoraWebCliente?x=10&y=20
```

El resultado en pantalla será:

```
    El resultado de la suma es: 30
    El primer parámetro es: 10
    El segundo parámetro es: 20
```

Caso 2. No pasamos los parámetros en la llamada http:

```
http://localhost:8080/CalculadoraWeb/calculadoraWebCliente
```

El resultado en pantalla será:

```
    Por favor, indique los números a sumar
```

Caso 3. Alguno de los parámetros es negativo:

```
http://localhost:8080/CalculadoraWeb/calculadoraWebCliente?x=10&y=-3
```

El resultado en pantalla será:

```
    El segundo sumando es negativo. Solo puedo sumar números positivos
```

4. Cliente para el servicio web Parte Metereológico

En las plantillas, disponéis de un war con la implementación del servicio web *Parte Metereologico*. Despliega dicho fichero: *ParteMetereologico.war* en el servidor de aplicaciones Glassfish para poder realizar el ejercicio.

Analizando su wsdl descubre qué dos operaciones ofrece el servicio

Registra el servicio en el gestor de servicios de Netbeans con el nombre de grupo

"expertoJava", prueba el servicio e implementa un cliente java Maven desde Netbeans para dicho servicio (el nombre del proyecto será Meteorologia-client).

Dicho cliente java debe realizar una llamada a las dos operaciones del servicio y mostrar en pantalla los resultados obtenidos. (1 punto)

Cuando crees el arquetipo de Maven, utiliza los siguientes nombres para:

- groupId: expertoJava
- artifactId: Meteorologia-Client

En este enlace podéis consultar un ejemplo de cómo de utilizar la clase *XMLGregorianCalendar*:
http://nachxs.wordpress.com/2010/04/26/como-usa-xmlgregoriancalendar-en-java/