Ejercicios de introducción a los Servicios Web

Índice

1. Invocación de Servicios Web

La página `http://www.xmethods.net` contiene una serie de Servicios Web de demostración. En la sección *XMethods Demo Services*, en la parte inferior de la página, tenemos una lista de servicios que podremos integrar en nuestras aplicaciones.

a) Pulsar sobre el servicio *Currency Exchange Rate* para obtener información sobre él. Esto nos llevará a una página con documentación sobre este servicio, además de un enlace al documento WSDL que lo describe.

- Leer la documentación del servicio. ¿Qué función realiza?
- Descargar el documento WSDL y editarlo. ¿Qué operaciones nos ofrece el servicio? ¿Qué parámetros toman y qué nos devuelven?
- Si pulsamos sobre el link *Analyze WSDL*, junto al link al documento WSDL, podremos ver los datos que aporta este documento de forma desglosada. Podemos ver el nombre del servicio, el puerto, las operaciones y los parámetros de entrada y datos devueltos por cada una de ellas.
- Buscar en este documento la información sobre:
  - Nombre del servicio y puerto.
  - Operaciones que ofrece el servicio.
  - Parámetros y datos devueltos por las operaciones.

b) Vamos a invocar el servicio mediante el cliente de prueba genérico de Weblogic (*Weblogic Test Client*). Para ello deberemos poner en marcha el servidor de aplicaciones e ir a la siguiente dirección para acceder al cliente de prueba:

```
http://localhost:7001/wls_utc
```

Introducir la URL donde se encuentra el documento WSDL que describe el servicio y probar a invocarlo con diferentes parámetros.

- Veremos una interfaz web con la lista de las operaciones que ofrece el servicio, en este caso tenemos sólo la operación `getRate`. Deberemos proporcionar los parámetros de entrada que toma este método, que serán el país de origen y el país de destino para la conversión monetaria, y a continuación pulsar sobre el botón para invocar la operación.
- Si queremos convertir la divisa europea (euro) a la de EEUU (dólar), ¿qué operación y qué parámetros debemos proporcionar?
- Una vez invocada la operación nos mostrará el resultado devuelto y los mensajes SOAP que se han utilizado para llamar al servicio.

c) Observar los mensajes SOAP de petición y respuesta utilizados. Relacionar los elementos

de estos mensajes con los elementos del documento WSDL que describe la interfaz del servicio.

- ¿Qué elementos encontramos en el mensaje de petición?
- ¿Y en el de respuesta?

2. Cliente para servicio web Hola Mundo

Vamos a crear un cliente para un servicio web sencillo. Se trata de un *"Hola mundo!"* en forma de servicio. Podemos encontrar el documento WSDL que define este servicio en la dirección `http://jtech.ua.es/HolaMundo/wsdl/HolaMundoSW.wsdl`. Se pide:

a) Invocar el servicio utilizando el cliente de prueba de Weblogic. Observar los mensajes SOAP de petición y respuesta utilizados.

b) Hemos accedido al servicio utilizando una herramienta genérica para probar Servicios Web. Sin embargo, la ventaja que nos ofrecen los Servicios Web es que podremos integrarlos en cualquier aplicación cliente, independientemente de la plataforma y del lenguaje sobre el que la desarrollemos.

Vamos a invocar el servicio utilizando la aplicación cliente propia contenida en el proyecto `servcweb-sesion01-weblogic` de BEA Workshop. La aplicación se puede ejecutar utilizando el objetivo `run` de ant, o ejecutando la clase `Main` directamente desde el entorno como aplicación Java.

c) En la clase `MainDebug` tenemos un cliente alternativo en el que se muestran los mensajes SOAP de petición y respuesta que se intercambian para utilizar el servicio. Ejecutar dicha clase (o el objetivo `run-debug` de ant) y observar el contenido de estos mensajes. Relacionar el contenido de estos mensajes con el del documento WSDL que define el servicio.

d) Observar el código fuente de la clase `Main`. Relacionar los elementos que se utilizan en esta clase (objetos que se instancian, métodos invocados, etc) con los elementos del fichero WSDL.

e) Crear un cliente básico para este servicio utilizando Netbeans. El cliente deberá ser una aplicación Java que invoque el servicio y que muestre en la consola el resultado obtenido. Se creará como un proyecto Java de nombre `servcweb-sesion01-hola`.

3. Clientes para otros servicios web (*)

Vamos a crear una serie de clientes para diferentes servicios web utilizando Netbeans.

a) Desarrollar un cliente para el servicio de cambio de moneda (*Currency Exchange Rate*) de XMethods, dentro de un proyecto Java de nombre `servcweb-sesion01-currency`. Podemos la URL del documento WSDL del servicio desde la página de *XMethods*. Esta dirección es la siguiente:

```
http://www.xmethods.net/sd/2001/CurrencyExchangeService.wsdl
```

Generar el cliente para este servicio utilizando la librería JAX-WS. ¿Qué ocurre y a qué es debido? Seleccionar la opción adecuada para que el servicio se pueda generar correctamente.

Si nos fijamos en el documento WSDL o en las clases generadas veremos que el servicio tiene como nombre `CurrencyExchangeService` y el tipo de puerto al que vamos a acceder se llama `CurrencyExchangePortType`. Introducir en la clase principal de la aplicación el código que invoque la operación `getRate()` del servicio y muestre el resultado en la consola. Esta operación toma dos parámetros: el país de origen y el de destino (por ejemplo `"euro"` o `"usa"`). Ejecutar el cliente y comprobar que funciona correctamente.

b) Vamos a utilizar los servicios web que ofrece Google, que nos permitirán integrar su motor de búsqueda en nuestras aplicaciones. Podemos encontrar el documento WSDL que define el servicio en la siguiente dirección:

```
http://api.google.com/GoogleSearch.wsdl
```

Generar el cliente del servicio e invocar el método de búsqueda `doGoogleSearch`, dentro de un proyecto Java de nombre `servcweb-sesion01-google`. Podemos invocarlo con los siguientes parámetros:

```
GoogleSearchResult result =
    googleSearchPort.doGoogleSearch(
            "iuH+5SdQFHIYv18KNcIjkg6RvfIUNYc6",
            "J2EE", 0, 10, true, "", false,
            "", "UTF8", "UTF8");
```

Donde el primer parámetro es un identificador del usuario que accede al servicio, y el segundo es la cadena de búsqueda. El resto de parámetros se utilizan para tratar cuestiones como la paginación de los resultados o su codificación. Como resultado nos devuelve un objeto `GoogleSearchResult` del cual podremos obtener la lista de páginas encontradas. Mostrar de cada una de ellas su título, su descripción (*snippet*) y su URL.

c) Vamos a realizar un cliente que acceda a los Servicios Web que ofrece *Amazon.com* para realizar búsquedas de productos en esta tienda, dentro de un proyecto Java de nombre `servcweb-sesion01-amazon`. Este es un servicio bastante más complejo que los

anteriores, por lo que necesitaremos documentación adicional para crear nuestro cliente. Podemos obtener documentación sobre estos servicios en la dirección:

```
http://aws.amazon.com
```

Para utilizar estos servicios necesitaremos registrarnos como desarrolladores, y obtener un código (*token*) que nos identifique cuando utilizamos los servicios de *Amazon*. Este código puede ser obtenido en la página anterior.

Nota:

Para las pruebas que vamos a realizar no es necesario obtener dicho código, ya que a continuación se proporciona uno en el fragmento de código de ejemplo.

Generar el cliente del servicio Amazon a partir del documento WSDL que lo define. Vamos a utilizar el servicio *Amazon E-Commerce Service*, cuyo documento WSDL puede ser consultado directamente en la dirección:

```
http://webservices.amazon.com/AWSECommerceService/AWSECommerceService.wsdl
```

Invocar desde el cliente la operación `itemSearch` para hacer una búsqueda de artículos. Concretamente los artículos que buscaremos serán DVDs, y realizaremos una búsqueda por director. Dado que la API de Amazon es bastante compleja, a continuación se muestra el código que deberíamos utilizar para invocar esta operación (se proporciona un *token* de desarrollador para que no sea necesario registrarse en la web):

```
String marketplaceDomain = null;
String awsAccessKeyId = "15JCR9XWMPTGV91JC1R2";
String subscriptionId = null;
String associateTag = null;
String xmlEscaping = null;
String validate = null;
ItemSearchRequest shared = null;
List<ItemSearchRequest> request = null;
Holder<OperationRequest> operationRequest = null;
Holder<List<Items>> items = new Holder<List<Items>>();

shared = new ItemSearchRequest();
shared.setSearchIndex("DVD");
shared.setDirector("Kubrick");
shared.getResponseGroup().add("Offers");
shared.getResponseGroup().add("Medium");

port.itemSearch(marketplaceDomain, awsAccessKeyId,
                subscriptionId, associateTag, xmlEscaping,
                validate, shared, request, operationRequest, items);

List<Items> listaResultados = items.value;
```

```
for(Items resultado: listaResultados) {
    List<Item> listaArticulos = resultado.getItem();
    for(Item articulo: listaArticulos) {
        ItemAttributes atributos = articulo.getItemAttributes();
        System.out.println("Titulo: " + atributos.getTitle());

        List<String> directores = atributos.getDirector();
        for(String director: directores) {
            System.out.print(director + "; ");
        }

        System.out.println("Precio: " +
            (atributos.getListPrice()!=null?
                atributos.getListPrice().getFormattedPrice():
                "no disponible"));
    }
}
```

Ejecutar la aplicación y comprobar que obtiene correctamente la lista de películas.