Ejercicios de introducción a los Servicios Web

Índice

1. Cliente para servicio web Hola Mundo

Vamos a probar 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) Observar el documento WSDL e identificar los diferentes elementos que contiene. ¿Qué operaciones ofrece el servicio? ¿Qué datos de entrada y de salida tiene cada una de ellas?.

b) Ejecutar el servicio mediante *Web Services Explorer*. Observar los mensajes SOAP utilizados para la invocación del servicio.

c) Crear en Eclipse un proyecto Java `servcweb-sesion01-hola`, y crear en él un cliente para el servicio.

2. Clientes para servicios web de la UA (*)

La Universidad de Alicante ofrece una serie de servicios web que nos dan acceso a información sobre las asignaturas, sus horarios, etc. Vamos a crear un cliente para acceder a estos servicios. Se pide:

a) Podemos consultar la documentación de los servicios y tener acceso a sus documentos WSDL en la web `http://cv1.cpd.ua.es/ws/`. Vamos a abrir el documento WSDL correspondiente a los servicios de dominio público (se encuentra en la sección *Serv. Informática*) de la web. ¿Qué tipo de codificación utiliza este servicio? ¿Esta codificación está soportada por los estándares actuales de servicios web (WS-I)? ¿Estará soportada por JAX-WS? Si no fuese así, ¿qué tendríamos que hacer si queremos crear un cliente para el servicio?

b) Probar el servicio mediante *Web Services Explorer*. Podemos invocar por ejemplo la operación `wshorarios`. Los códigos de las asignaturas figuran en la web de la Universidad, podríamos utilizar por ejemplo el código `9181` que corresponde a la asignatura *Ingeniería del Software 2*. Observar el contenido de los mensajes SOAP de petición y respuesta.

c) Generar el cliente del servicio en un nuevo proyecto Java al que llamaremos `servcweb-sesion01-ua`, e invocar dentro de él el método para obtener los horarios de una asignatura (`wshorarios`).

3. Servicios web de Amazon (*)

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 en Netbeans 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:

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

Importante

Este servicio web no es compatible con la API JAX-RPC 1.1 que usa Axis, por lo que no podremos invocarlo con estas librerías.

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"));
    }
}
```

¡Cuidado!

Seleccionar `imports` incorrectos al introducir este código puede hacer que no compile correctamente. Todos los artefactos que representan los objetos de Amazon pertenecerán al paquete en el que se ha generado el *stub*.

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