Cliente SOAP del servicio web de Amazon (bare parameter style)

Creación de los stubs

```
wsimport -p amazon -keep \
http://webservices.amazon.com/AWSECommerceService/AWSECommerceService.wsdl\
-b custom.xml
```

Fichero custom.xml

```
<jaxws:bindings
  wsdlLocation ="http://ecs.amazonaws.com/AWSECommerceService/AWSECommerceService.wsdl"
  xmlns:jaxws="http://java.sun.com/xml/ns/jaxws">
<jaxws:enableWrapperStyle>false</jaxws:enableWrapperStyle>
</jaxws:bindings>
```

Programa Java

```
package amazon;

import amazon.AWSECommerceService;
import amazon.AWSECommerceServicePortType;
import amazon.ItemSearchRequest;
import amazon.ItemSearchResponse;
import amazon.ItemSearch;
import amazon.Items;
import amazon.Item;
import amazon.AwsHandlerResolver;
import java.util.List;

class AmazonClientBareStyle
{
  public static void main(String[ ] args)
  {
    if (args.length < 2) {
       System.err.println("AmazonClientBareStyle <accessId> <secretKey>");
       return;
    }
    final String accessId = args[0];
    final String secretKey = args[1];

    AWSECommerceService service = new AWSECommerceService();
    service.setHandlerResolver(new AwsHandlerResolver(secretKey));
    AWSECommerceServicePortType port = service.getAWSECommerceServicePort();
    ItemSearchRequest request = new ItemSearchRequest();
    request.setSearchIndex("Books");
    request.setKeywords("data integration");
    ItemSearch itemSearch= new ItemSearch();
    itemSearch.setAWSAccessKeyId(accessId);
    itemSearch.setAssociateTag("ASSOCIATE-ID");
    itemSearch.getRequest().add(request);
    ItemSearchResponse response = port.itemSearch(itemSearch);
    List<Items> itemsList = response.getItems();

    for (Items next : itemsList)
        for (Item item : next.getItem())
            System.out.println(item.getItemAttributes().getTitle());
  }
}
```

Cliente SOAP del servicio web de Amazon (wrapped parameter style)

Creación de los stubs

```
wsimport -p amazon2 -keep \
http://webservices.amazon.com/AWSECommerceService/AWSECommerceService.wsdl
```

Programa Java

```
package amazon2;

import amazon2.AWSECommerceService;
import amazon2.AWSECommerceServicePortType;
import amazon2.ItemSearchRequest;
import amazon2.ItemSearch;
import amazon2.Items;
import amazon2.Item;
import amazon2.OperationRequest;
import amazon2.SearchResultsMap;
import amazon2.AwsHandlerResolver;
import javax.xml.ws.Holder;
import java.util.List;
import java.util.ArrayList;

class AmazonClientWrappedStyle
{
  public static void main(String[ ] args)
  {
    if (args.length < 2) {
       System.err.println("java AmazonClientWrappedStyle <accessId> <secretKey>");
       return;
    }
    final String accessId = args[0];
    final String secretKey = args[1];
    AWSECommerceService service = new AWSECommerceService();
    service.setHandlerResolver(new AwsHandlerResolver(secretKey));
    AWSECommerceServicePortType port = service.getAWSECommerceServicePort();
    ItemSearchRequest request = new ItemSearchRequest();
    request.setSearchIndex("Books");
    request.setKeywords("data integration");
    ItemSearch search = new ItemSearch();
    search.getRequest().add(request);
    search.setAWSAccessKeyId(accessId);
    search.setAssociateTag("ASSOCIATE-ID");
    Holder<OperationRequest> operationRequest = null;
    Holder<List<Items>> items = new Holder<List<Items>>();
    port.itemSearch(
       search.getMarketplaceDomain(),
       search.getAWSAccessKeyId(),
       search.getAssociateTag(),
       search.getXMLEscaping(),
       search.getValidate(),
       search.getShared(),
       search.getRequest(),
       operationRequest,
       items);

    Items retval = items.value.get(0);
    List<Item> item_list = retval.getItem();
    for (Item item : item_list)
       System.out.println(item.getItemAttributes().getTitle());
  }
}
```

Cliente REST del servicio web de Amazon

```
package restful;

import java.net.URL;
import java.net.URLConnection;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.ByteArrayInputStream;
import java.util.HashMap;
import java.util.Map;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

public class RestfulAmazon
{
  private static final String endpoint = "ecs.amazonaws.com";
  private static final String itemId = "0545010225"; // Harry Potter

  public static void main(String[ ] args)
  {
    if (args.length < 2) {
        System.err.println("RestfulAmazon <accessKeyId> <secretKey>");
        return;
    }

    new RestfulAmazon().lookupStuff(args[0].trim(), args[1].trim());
  }

  private void lookupStuff(String accessKeyId, String secretKey)
  {
    RequestHelper helper = new RequestHelper(endpoint, accessKeyId, secretKey);
    String requestUrl = null;
    String title = null;
    // Store query string params in a hash.
    Map<String, String> params = new HashMap<String, String>();
    params.put("Service", "AWSECommerceService");
    params.put("Version", "2009-03-31");
    params.put("Operation", "ItemLookup");
    params.put("ItemId", itemId);
    params.put("ResponseGroup", "Small");
    params.put("AssociateTag", "ASSOCIATE-ID");

    requestUrl = helper.sign(params);

    String response = requestAmazon(requestUrl);
    // The string "null" is returned before the XML document.
    String noNullResponse = response.replaceFirst("null", "");
    System.out.println("Raw xml:\n" + noNullResponse);
    System.out.println("Author: " + getAuthor(noNullResponse));
  }
```

```
  private String requestAmazon(String stringUrl)
  {
    String response = null;
    try {
       URL url = new URL(stringUrl);
       URLConnection conn = url.openConnection();
       conn.setDoInput(true);
       BufferedReader in = new BufferedReader(new InputStreamReader(conn.getInputStream()));
       String chunk = null;
       while ((chunk = in.readLine()) != null) response += chunk;
       in.close();
    } catch (Exception e) {
       throw new RuntimeException("Arrrg! " + e);
    }
    return response;
  }

private String getAuthor(String xml)
{
  String author = null;
  try {
    ByteArrayInputStream bais = new ByteArrayInputStream(xml.getBytes());
    DocumentBuilderFactory fact = DocumentBuilderFactory.newInstance();
    fact.setNamespaceAware(true);
    DocumentBuilder builder = fact.newDocumentBuilder();
    Document doc = builder.parse(bais);
    NodeList results = doc.getElementsByTagName("Author");
    for (int i = 0; i < results.getLength(); i++) {
        Element e = (Element) results.item(i);
        NodeList nodes = e.getChildNodes();
        for (int j = 0; j < nodes.getLength(); j++) {
            Node child = nodes.item(j);
            if (child.getNodeType() == Node.TEXT_NODE)
                author = child.getNodeValue();
        }
     }
  } catch (Exception e) {
      throw new RuntimeException("Xml bad!", e);
  }
  return author;
}


}
```

```
package restful;

import java.io.UnsupportedEncodingException;
import java.net.URLDecoder;
import java.net.URLEncoder;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.SortedMap;
import java.util.TimeZone;
import java.util.TreeMap;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import org.apache.commons.codec.binary.Base64;

public class RequestHelper
{
  private static final String utf8 = "UTF-8";
  private static final String hmacAlg = "HmacSHA256";
  private static final String requestUri = "/onca/xml";
  private static final String requestMethod = "GET";
  private String endpoint = null;
  private String accessKeyId = null;
  private String secretKey = null;
  private SecretKeySpec secretKeySpec = null;

  private Mac mac = null; public RequestHelper(String endpoint, String accessKeyId, String secretKey)
  {
    if (endpoint == null || endpoint.length() == 0)
        throw new RuntimeException("The endpoint is null or empty.");
    if (null == accessKeyId || accessKeyId.length() == 0)
        throw new RuntimeException("The accessKeyId is null or empty.");
    if (null == secretKey || secretKey.length() == 0)
        throw new RuntimeException("The secretKey is null or empty.");
    this.endpoint = endpoint.toLowerCase();
    this.accessKeyId = accessKeyId;
    this.secretKey = secretKey;
    try {
        byte[ ] secretKeyBytes = this.secretKey.getBytes(utf8);
        this.secretKeySpec = new SecretKeySpec(secretKeyBytes, hmacAlg);
        this.mac = Mac.getInstance(hmacAlg);
        this.mac.init(this.secretKeySpec);
    } catch(Exception e) { throw new RuntimeException(e); }
  }

  public String sign(Map<String, String> params)
  {
    params.put("AWSAccessKeyId", this.accessKeyId);
    params.put("Timestamp", this.timestamp());
    // The parameters need to be processed in lexicographical order, with sorting on the first byte
    SortedMap<String, String> sortedParamMap = new TreeMap<String, String>(params);
    // Ensure canonical form of the query string, as Amazon REST is fussy.
    String canonicalQS = this.canonicalize(sortedParamMap);
    // Prepare the signature with grist for the mill.
    String toSign = requestMethod + "\n" + this.endpoint + "\n" + requestUri + "\n" + canonicalQS;
    String hmac = this.hmac(toSign);
    String sig = null;
    try {
        sig = URLEncoder.encode(hmac, utf8);
    } catch(UnsupportedEncodingException e) { System.err.println(e); }
    String url = "http://" + this.endpoint + requestUri + "?" + canonicalQS + "&Signature=" + sig;
    return url;
  }
```

```
public String sign(String queryString)
{
  Map<String, String> params = this.createParameterMap(queryString);
  return this.sign(params);
}

private String hmac(String stringToSign)
{
  String signature = null;
  byte[ ] data;
  byte[ ] rawHmac;
  try {
      data = stringToSign.getBytes(utf8);
      rawHmac = mac.doFinal(data);
      Base64 encoder = new Base64();
      signature = new String(encoder.encode(rawHmac));
  } catch (UnsupportedEncodingException e) {
      throw new RuntimeException(utf8 + " is unsupported!", e);
  }
  return signature;
}

// Amazon requires an ISO-8601 timestamp.
private String timestamp()
{
  String timestamp = null;
  Calendar cal = Calendar.getInstance();
  DateFormat dfm = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss'Z'");
  dfm.setTimeZone(TimeZone.getTimeZone("GMT"));
  timestamp = dfm.format(cal.getTime());
  return timestamp;
}

private String canonicalize(SortedMap<String, String> sortedParamMap)
{
  if (sortedParamMap.isEmpty()) return "";
  StringBuffer buffer = new StringBuffer();
  Iterator<Map.Entry<String, String>> iter = sortedParamMap.entrySet().iterator();
  while (iter.hasNext()) {
      Map.Entry<String, String> kvpair = iter.next();
      buffer.append(encodeRfc3986(kvpair.getKey()));
      buffer.append("=");
      buffer.append(encodeRfc3986(kvpair.getValue()));
      if (iter.hasNext()) buffer.append("&");
  }
  return buffer.toString();
}

// Amazon requires RFC 3986 encoding, which the URLEncoder may not get right.
private String encodeRfc3986(String s)
{
  String out;
  try {
      out = URLEncoder.encode(s, utf8)
              .replace("+", "%20")
              .replace("*", "%2A")
              .replace("%7E", "~");
  } catch (UnsupportedEncodingException e) {
      out = s;
  }
  return out;
}
```

```
  private Map<String, String> createParameterMap(String queryString)
  {
    Map<String, String> map = new HashMap<String, String>();
    String[ ] pairs = queryString.split("&");
    for (String pair : pairs) {
        if (pair.length() < 1) continue;
        String[ ] tokens = pair.split("=", 2);
        for(int j = 0; j < tokens.length; j++) {
            try {
                tokens[j] = URLDecoder.decode(tokens[j], utf8);
            } catch (UnsupportedEncodingException e) { }
        }
        switch (tokens.length) {
            case 1: {
                if (pair.charAt(0) == '=') map.put("", tokens[0]);
                else map.put(tokens[0], "");
                break;
            }
            case 2: {
                map.put(tokens[0], tokens[1]);
                break;
            }
        }
    }
    return map;
  }
}
```

Cliente REST del servicio web de Amazon: JAX-B (Java XML Bindings)

Esquema XML

http://webservices.amazon.com/AWSECommerceService/AWSECommerceService.xsd

NOTA: Reemplazar el identificador “xs” por “xsd”

Creación de clases auxiliares:

```
xjc -p restful2 amazon.xsd
```

Código en Java (igual que el cliente anterior en su mayor parte):

```
...
  private String getAuthor(String xml)
  {
    String author = null;
    try {
       // Create an XML Schema object
       final String fileName = "amazon.xsd"; // downloaded XML Schema
       final String schemaUri = XMLConstants.W3C_XML_SCHEMA_NS_URI;
       SchemaFactory factory = SchemaFactory.newInstance(schemaUri);
       Schema schema = factory.newSchema(new StreamSource(fileName));
       // Create a JAX-B context for unmarshaling
       JAXBContext ctx = JAXBContext.newInstance(ItemLookupResponse.class);
       Unmarshaller um = ctx.createUnmarshaller();
       um.setSchema(schema);
       // Generate a Java ItemSearchResponse instance.
       ItemLookupResponse ilr = (ItemLookupResponse)
       um.unmarshal(new ByteArrayInputStream(xml.getBytes()));
       // Use the standard POJO idiom to extract the author.
       List<Items> itemsList = ilr.getItems(); // list of lists
       for (Items items : itemsList) { // outer list
          List<Item> list = items.getItem(); // inner list
          for (Item item : list) { // items in inner list
             ItemAttributes attributes = item.getItemAttributes();
             List<String> authors = attributes.getAuthor(); // could be several
             author = authors.get(0); // in this case, only one
          }
       }
    } catch(JAXBException e ) {
       throw new RuntimeException(e);
    } catch(Exception e) {
       throw new RuntimeException(e);
    }
    return author;
  }
}
```