

4.2 SOAPメッセージを受信するSOAPサービスの構築

まずはクライアントアプリケーションからメッセージを受取り、処理することのできるSOAPサービスを構築します。SOAPメッセージングを使用したSOAPサービス用のプログラムは、SOAP-RPCと異なり、SOAPの構造を意識した記述と、XML文書を解析するコードが必要になります。

SOAPメッセージングを使用するサービスプログラムの処理の流れ

Apache Axis2では、直接XMLを操作するのではなくAxisのXMLデータモデルであるAXIOMを使用してSOAPメッセージの操作を行います。

AXIOMを処理する場合には、一つのorg.apache.axiom.om.OMElementを受け取り、org.apache.axiom.om.OMElementを返すメソッドを用意します。

```
public OMElement method(OMElement element);
```

このときメソッド名は、メッセージのBody要素の子要素名と同じにします。Apache Axis2では、SOAPメッセージのBody要素の子要素名がサービスのメソッド名に対応します。

受け取るOMElementは受信したSOAPメッセージのBody要素の子要素です。

返すOMElementも、返信するSOAPメッセージのBody要素の子要素です。

ここでは、次のようなSOAPメッセージの受信を想定しています。

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <getBookInfo xmlns="urn:ICECBookData2">
      <title>書籍XML基礎</title>
      <author>ICEC</author>
    </getBookInfo>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAPメッセージングを使用するサービスプログラムでは、次のような内容を記述します。

1. 公開しているメソッドの引数として、OMElementオブジェクトを取得する。(9～11行)
2. 処理の実装 (13～15行)
3. 返信用SOAPメッセージのBody要素の直接の子要素であるBook要素、およびその子要素を構成する。(17～21行)
4. 最後に返信用SOAPメッセージのBody要素の子要素を返却します。このメソッドが終了した時点でレスポンスのSOAPメッセージがクライアント側に送信されます。(22行)

(※行数はサンプルコードの行数です。)

■ サンプルコード

ファイル名 : BookDataService2.java

```
01: import javax.xml.stream.XMLStreamException;
02:
03: import org.apache.axiom.om.OMAbstractFactory;
04: import org.apache.axiom.om.OMElement;
05: import org.apache.axiom.om.OMFactory;
06: import org.apache.axiom.om.util.ElementHelper;
07:
08: public class BookDataService2 {
09:     public OMElement getBookInfo(OMElement element) throws XMLStreamException {
10:         element.build();
11:         element.detach();
12:
13:         OMElement omTitle = ElementHelper.getChildWithName(element, "title");
14:         String title = omTitle.getText();
15:         String ret = (title.equals("書籍XML基礎") ? "13000" : "0");
16:
17:         OMFactory factory = OMAbstractFactory.getOMFactory();
18:         OMElement book = factory.createOMElement("Book", "", "");
19:         OMElement price = factory.createOMElement("Price", "", "");
20:         factory.createOMText(price, ret);
21:         book.addChild(price);
22:         return book;
23:     }
24: }
```

■ サンプルコードの解説

◇クラスのインポート

```
01: import javax.xml.stream.XMLStreamException;
02:
03: import org.apache.axiom.om.OMAbstractFactory;
04: import org.apache.axiom.om.OMElement;
05: import org.apache.axiom.om.OMFactory;
06: import org.apache.axiom.om.util.ElementHelper;
```

SOAPメッセージングの場合、XMLから所定のデータを取得したり、あるいは返却用のXMLを構築するプログラミングを行う必要があります。その際に使用する、AXIOMプログラミングでのノード操作のための基本的なクラスがOMElementクラスです。このクラスはW3C DOMによく似たインターフェースを実装しているとともに、ノード操作のための便利なメソッドが定義されていますので、OMElementクラスを使用し、XMLのノード操作を有効に進めることができます。

◇受信したOMElementオブジェクトの取得

```
09: public OMElement getBookInfo(OMElement element) throws XMLStreamException {
10:     element.build();
11:     element.detach();
```

メソッド名は、SOAPメッセージのBody要素の子要素名と同じです。
メソッドの引数として受け取ったOMElementオブジェクトは、受信したSOAPメッセージのgetBookInfo要素を表現しています。このとき11行目のdetachメソッドで、getBookInfo要素は親要素（Book要素）から切り離されます。

◇処理の実装

```
13:     OMElement omTitle = ElementHelper.getChildWithName(element, "title");
```

elementはgetBookInfo要素を参照しているため、その子要素の中からtitle要素を取得しています。

```
14:     String title = omTitle.getText();
15:     String ret = (title.equals("書籍XML基礎") ? "13000" : "0");
```

title要素の文字列を取得し、「書籍XML基礎」と等しければ「13000」を、そうでなければ「0」をレスポンス用文字列に設定しています。

◇返信用SOAPメッセージのBody要素以下を構成し、返却する

```
17:     OMFactory factory = OMAbstractFactory.getOMFactory();
18:     OMElement book = factory.createOMElement("Book", "", "");
19:     OMElement price = factory.createOMElement("Price", "", "");
20:     factory.createOMText(price, ret);
21:     book.addChild(price);
22:     return book;
```

レスポンスの文字列が「13000」であるとき、レスポンスのXML（Body要素の子要素）を次の構造にします。

```
<Book>
  <Price>13000</Price>
</Book>
```

OMFactoryオブジェクトのcreateOMElementメソッドを使用して、Book、Price要素を作成します。

作成したBook要素の子要素としてPrice要素を設定し、Price要素の内容文字列としてレスポンスの文字列を設定します。