DotNetInteropArrays

AXIS and .NET can exchange Arrays. Arrays of primitives, or Arrays of complex types.

First thing to do is stay away from `SOAPENC:arrayType` since we all listen to WS-I. We'll do doc/literal encoding. Agreed?

Ok, then Within the WSDL, in the <types> section, specify a type or element that wraps an array, using an element with maxOccurs != 1, like so:

```
 <s:complexType name="Container">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="param1" nillable="true" type="s:string" />
    <s:element minOccurs="0" maxOccurs="unbounded" name="param2" type="s:string" />
  </s:sequence>
 </s:complexType>
```

The param2 above can be of any type (simple or complex or XSD primitive).

Then, use WSDL2Java to generate the Java classes for this. The generated Java code will look like the following:

```
class Container{
      String param1;
      String[] param2;

      public String getParam1() {
          return param1;
      }
      public void setParam1(String value) {
          param1 = value;
      }

      public String getParam2(index i) {
          return param2[i];
      }
      public void setParam2(index i, String value) {
          param2[i] = value;
      }
      public String[] getParam2() {
          return param2;
      }
      public void setParam2(String[] value) {
          param2 = value;
      }
  }
```

The XML on the wire will be:

```
 <Container>
   <param1> foo</param1>
   <param2>bar</param2>
   <param2>blah</param2>
       ...
 </Container>
```

And the .NET client will be completely happy with that.

On the other hand if you structure your WSDL to use a wrapper type (ArrayOf*_Xxx) for the array, like so:

```
 <s:complexType name="Container">
   <s:sequence>
     <s:element minOccurs="1" maxOccurs="1" name="param1" nillable="true" type="s:string" />
     <s:element minOccurs="1" maxOccurs="1" name="wrapper" nillable="true" type="tns:ArrayOfString" />
   </s:sequence>
 </s:complexType>
 <s:complexType name="ArrayOfString">
   <s:sequence>
     <s:element minOccurs="0" maxOccurs="unbounded" name="param2" type="s:string" />
   </s:sequence>
 </s:complexType>
```

(the ArrayOf_*'Xxx name is actually not significant, just the structure is important.) ... then the generated Java code will look like:

```
class Container {
    String param1;
    ArrayOfString wrapper;

    public String getParam1() {
        return param1;
    }
    public void setParam1(String value) {
        param1 = value;
    }
    public String getWrapper() {
        return wrapper;
    }
    public void setWrapper(ArrayOfString value) {
        wrapper= value;
    }
}

class ArrayOfString{
    String[] param2;

    public String getParam2(index i) {
        return param2[i];
    }
    public void setParam2(index i, String value) {
        param2[i] = value;
    }
    public String[] getParam2() {
        return param2;
    }
    public void setParam2(String[] value) {
        param2 = value;
    }
}
```

... and the XML on the wire will be like this:

```
<Container>
  <param1> foo</param1>
  <wrapper>
      <param2>bar</param2>
      <param2>blah</param2>
      ...
  </wrapper>
</Container>
```

and .NET will be happy with that also.

Using the wrapper type specified in the WSDL, the programming model in the generated Java/AXIS code explicitly exposes the ArrayOf'_*Xxx type. On the other hand, with .NET, the programming model is the same: both styles of WSDL will generate simple arrays like MyType[]. What varies on the .NET side is the metadata attached to the artifacts in the form of inline attributes.

.NET can go either way. It takes its cue from the WSDL.

So you can do whatever.

At one point, there was a bug in AXIS dealing with an operation that returns an array directly, rather than a structure containing an array (ie, an operation like `String[] getInfo()` rather than

```
Container getInfo()
```

). This may still be present in AXIS V1.2RC3. Dims?

All of the above supposes that you are doing *WSDL First_'. This may not be the case. If not, then you should structure your Java code so that it reflects what AXIS would generate if you did actually use WSDL2Java to generate the AXIS code. This means JavaBean type classes, with setters and getters as above for the arrays. Be sure to get the indexed setter+getter as well as the array setter+getter.

When you then generate WSDL from such Java classes, you will get what is reflected in the above examples.