WSDL Binding

Sang Shin
Michèle Garoche
www.javapassion.com
"Learning is fun!"

Agenda

- Binding Element
- SOAP Binding
  - style=”rpc” use=”literal”
  - style=”document” use=”literal”
  - style=”rpc” use=”encoded”
  - style=”document” use=”encoded”
  - When to Use What?

Binding Element

Binding Element

- Defines protocol details and message format for *operations* and *messages* defined by a particular *portType*
- Specify one protocol out of
  - SOAP (SOAP over HTTP, SOAP over SMTP)
  - HTTP GET/POST
- Provides extensibility mechanism
  - Can includes binding extensibility elements
  - Binding extensibility elements are used to specify the concrete grammar

Binding Element Syntax

```
<wsdl:definitions .... >
   ...

   <wsdl:binding name="nmtoken" type="qname"> *
      <-- extensibility element per binding --> *

      <wsdl:operation name="nmtoken"> *
         <-- extensibility element per operation --> *

         <wsdl:input name="nmtoken"? > ?
            <-- extensibility element per input -->
         </wsdl:input>
         <wsdl:output name="nmtoken"? > ?
            <-- extensibility element per output --> *
         </wsdl:output>
         <wsdl:fault name="nmtoken"> *
            <-- extensibility element per fault --> *
         </wsdl:fault>

      </wsdl:operation>

   </wsdl:binding>
</wsdl:definitions>
```

SOAP Binding

SOAP Binding Extension

- WSDL includes binding for SOAP 1.1/1.2 endpoints and supports:
  - Indication of binding to SOAP as a protocol
  - Address for SOAP endpoint
  - The URI for SOAPAction HTTP header (applies only for HTTP binding of SOAP)
  - List of definitions for Headers for SOAP envelope
- “soap” namespace
  - *xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"* (for using SOAP 1.1)
  - *xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"* (for using SOAP 1.2)

SOAP Binding Extensions Syntax

```
<definitions .... >
  <binding .... >
    <soap:binding style="rpc|document" transport="uri">
    <operation .... >
      <soap:operation soapAction="uri"? style="rpc|document"?>?
      <input>
        <soap:body parts="nmtokens"? use="literal|encoded" encodingStyle="uri-list"? namespace="uri"?>
        <soap:header message="qname" part="nmtoken" use="literal|encoded"
                encodingStyle="uri-list"? namespace="uri"?>*
          <soap:headerfault message="qname" part="nmtoken" use="literal|encoded"
                    encodingStyle="uri-list"? namespace="uri"?/>*
        <soap:header>
      </input>
      <output>
        <soap:body parts="nmtokens"? use="literal|encoded" encodingStyle="uri-list"? namespace="uri"?>
        <soap:header message="qname" part="nmtoken" use="literal|encoded"
                encodingStyle="uri-list"? namespace="uri"?>*
          <soap:headerfault message="qname" part="nmtoken" use="literal|encoded"
                    encodingStyle="uri-list"? namespace="uri"?/>*
        <soap:header>
      </output>
      <fault>*
        <soap:fault name="nmtoken" use="literal|encoded" encodingStyle="uri-list"? namespace="uri"?>
       </fault>
    </operation>
  </binding>

  <port .... >
    <soap:address location="uri"/>
  </port>
</definitions>
```

soap:binding

```
<definitions .... >
   <binding .... >
      <soap:binding transport="uri"? style="rpc|document"?>
   </binding>
</definitions>
```

- Must be present when using SOAP binding
- *style* attribute applies to each contained operation (default: *document*) unless it is overidden by operation specific *style* attribute
- *transport* attribute indicates which transport to use
  - http://schemas.xmlsoap.org/soap/http (for HTTP)
  - http://schemas.xmlsoap.org/soap/smtp (for SMTP)

soap:operation

```
<definitions .... >
   <binding .... >
      <operation .... >
        <soap:operation soapAction="uri"? style="rpc|document"?>?
      </operation>
   </binding>
</definitions>
```

- *style* attribute indicates whether the operation is RPC-oriented (messages containing parameters and return values) or document-oriented (message containing document(s))
  - Affects the way in which the Body of the SOAP message is constructed on the wire
- *soapAction* attribute specifies the value of the *SOAPAction* header for this operation

soap:body

```
<definitions .... >
   <binding .... >
      <operation .... >
        <input>
           <soap:body parts="nmtokens"? use="literal|encoded"?
                      encodingStyle="uri-list"? namespace="uri"?>
        </input>
        <output>
           <soap:body parts="nmtokens"? use="literal|encoded"?
                      encodingStyle="uri-list"? namespace="uri"?>
        </output>
      </operation>
   </binding>
</definitions>
```

soap:body

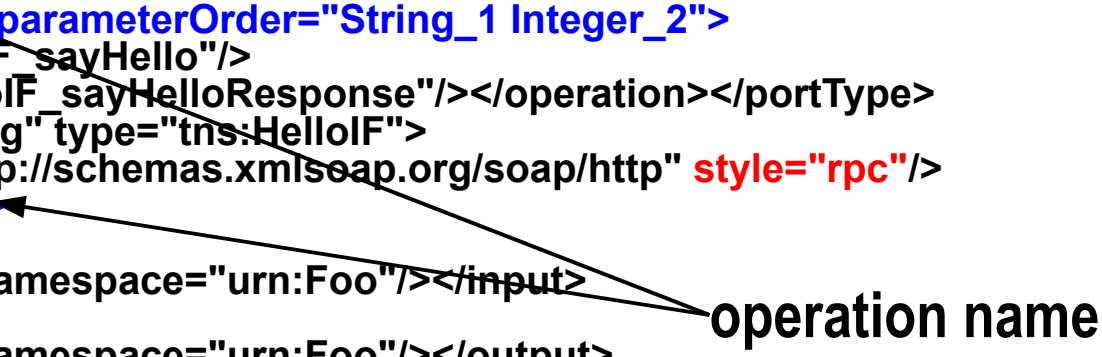
- Specifies how the message parts appear inside the SOAP Body element
  - Provides information on how to assemble the different message parts inside the Body element
- Used in both RPC-oriented and document-oriented messages
  - Which one to use is determined via *style* attribute of soap:binding or soap:operation elements

soap:body for RPC style

- WSDL document
  - The operation name of WSDL document is used to name the wrapper element (immediate child element under <soap:Body> element)
    - Name of the wrapper element is a method
  - Each part is a parameter or a return value and appears inside a wrapper element within the <soap:Body>
- SOAP message:
  - Contents of the Body are formatted as a struct
  - Parts are arranged in the same order as the parameters of the call

MyHelloServiceRpcLiteral.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="urn:Foo"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" name="MyHelloService"
   targetNamespace="urn:Foo">
  <types/>
  <message name="HelloIF_sayHello">
    <part name="String_1" type="xsd:string"/>
    <part name="Integer_2" type="xsd:int"/></message>
  <message name="HelloIF_sayHelloResponse">
    <part name="result" type="xsd:string"/></message>
  <portType name="HelloIF">
    <operation name="sayHello" parameterOrder="String_1 Integer_2">
      <input message="tns:HelloIF_sayHello"/>
      <output message="tns:HelloIF_sayHelloResponse"/></operation></portType>
  <binding name="HelloIFBinding" type="tns:HelloIF">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
    <operation name="sayHello">
      <input>
        <soap:body use="literal" namespace="urn:Foo"/></input>
      <output>
        <soap:body use="literal" namespace="urn:Foo"/></output>
      <soap:operation soapAction=""/></operation>
   </binding>
  <service name="MyHelloService">
    <port name="HelloIFPort" binding="tns:HelloIFBinding">
      <soap:address xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
      location="http://localhost:8080/hello-jaxrpc/hello"/></port></service></definitions>
```



SOAP Request Message: RPC, Literal

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:n="urn:Foo"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
    <n:sayHello>
      <String_1>MyRpcLiteralMessage</String_1>
      <Integer_2>79</Integer_2>
    </n:sayHello>
  </soap:Body>
</soap:Envelope>
```

```
<part name="String_1"
    type="xsd:string"/>
<part name="Integer_2" type="xsd:int"/>
```

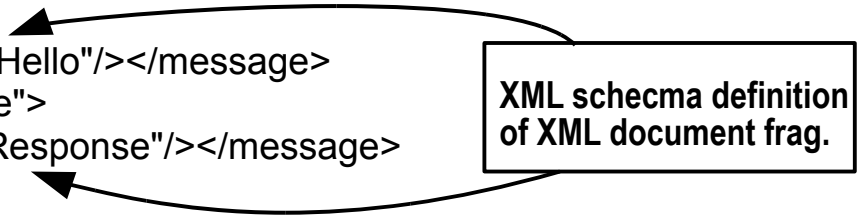
```
<operation name="sayHello" parameterOrder="String_1 Integer_2">
```

soap:body for Document style

- WSDL document:
  - Each <message> has single <part> element
  - The element attribute of <part> refers to schema definition of XML document fragment, which is defined inside <types>
- SOAP message:
  - SOAP Body element contains an XML document fragment (document)
    - Ex) Purchase order XML document fragment
  - There are no wrappers

MyHelloServiceDocLiteral.wsdl(1)

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="urn:Foo"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
      name="MyHelloService"targetNamespace="urn:Foo">
 <types>
  <schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:soap11
    enc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" targetNamespace="urn:Foo">
    <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
    <complexType name="sayHello">
     <sequence>
      <element name="String_1" type="string" nillable="true"/>
      <element name="Integer_2" type="int" nillable="true"/></sequence></complexType>
    <complexType name="sayHelloResponse">
     <sequence>
      <element name="result" type="string" nillable="true"/></sequence></complexType>
    <element name="sayHello" type="tns:sayHello"/>
    <element name="sayHelloResponse" type="tns:sayHelloResponse"/></schema></types>
 <message name="HelloIF_sayHello">
  <part name="parameters" element="tns:sayHello"/></message>
 <message name="HelloIF_sayHelloResponse">
  <part name="result" element="tns:sayHelloResponse"/></message>
```



SOAP Request Message: Doc, Literal

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:tns="urn:Foo"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <tns:sayHello>
      <String_1>MyDocLiteralMessage</String_1>
      <Integer_2>78</Integer_2>
    </tns:sayHello>
  </soap:Body>
</soap:Envelope>
```

```
<complexType name="sayHello">
    <sequence>
      <element name="String_1" type="string" nillable="true"/>
      <element name="Integer_2" type="int" nillable="true"/>
    </sequence>
</complexType>
```

use attribute of soap:body

- *use="literal|encoded"*
- *literal*
  - parts define the concrete schema of the message
  - XML document fragment can be validated against its XML schema
- *encoded*
  - Indicates whether the message parts are encoded using some encoding rules

use attribute of soap:body

- use=”literal”
  - each part references a concrete schema definition using either the *element* or *type* attribute (WS-I profile says use *element*)
  - *element* attribute
    - Document style: the element referenced by the part will appear directly under the Body element
    - RPC style: the element referenced by the part will appear under an accessor element named after the message part
  - *type* attribute
    - the type referenced by the part becomes the schema type of the enclosing element

use attribute of soap:body

- use=”encoded”
  - each message part references an abstract type using the *type* attribute
  - abstract types are used to produce a concrete message by applying an encoding specified by the *encodingStyle* attribute
  - part names, types and value of the namespace attribute are all inputs to the encoding

Possible Style/Use Combinations

- style=”rpc” and use=”encoded”
- style=”rpc” and use=” literal”
- style=”document” and use=”encoded”
- style=”document” and use=”literal”

style=”rpc”
use=”literal”

SOAP Binding

MyHelloServiceRpcLiteral.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="urn:Foo"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" name="MyHelloService"
   targetNamespace="urn:Foo">
  <types/>
  <message name="HelloIF_sayHello">
    <part name="String_1" type="xsd:string"/>
    <part name="Integer_2" type="xsd:int"/></message>
  <message name="HelloIF_sayHelloResponse">
    <part name="result" type="xsd:string"/></message>
  <portType name="HelloIF">
    <operation name="sayHello" parameterOrder="String_1 Integer_2">
      <input message="tns:HelloIF_sayHello"/>
      <output message="tns:HelloIF_sayHelloResponse"/></operation></portType>
  <binding name="HelloIFBinding" type="tns:HelloIF">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
    <operation name="sayHello">
      <input>
        <soap:body use="literal" namespace="urn:Foo"/></input>
      <output>
        <soap:body use="literal" namespace="urn:Foo"/></output>
      <soap:operation soapAction=""/></operation>
   </binding>
  <service name="MyHelloService">
    <port name="HelloIFPort" binding="tns:HelloIFBinding">
      <soap:address xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        location="http://localhost:8080/hello-jaxrpc/hello"/></port></service></definitions>
```

SOAP Request Message: RPC, Literal

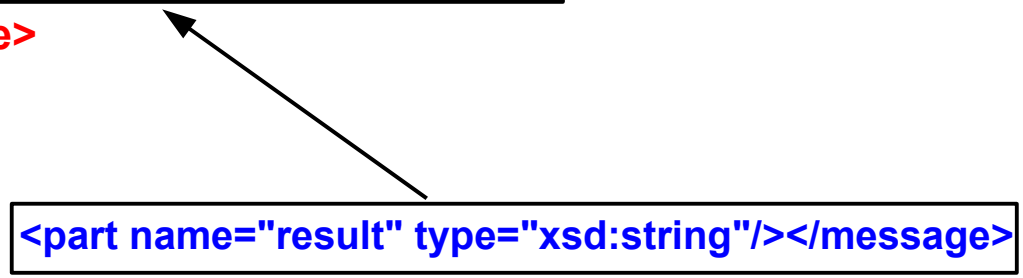
```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:n="urn:Foo"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
    <n:sayHello>
      <String_1>MyRpcLiteralMessage</String_1>
      <Integer_2>79</Integer_2>
    </n:sayHello>
  </soap:Body>
</soap:Envelope>
```

```
<part name="String_1"
    type="xsd:string"/>
<part name="Integer_2" type="xsd:int"/>
```

```
<operation name="sayHello" parameterOrder="String_1 Integer_2">
```

SOAP Response Message: Rpc, Literal

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope
    xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:ns0="urn:Foo">
   <env:Body>
      <ns0:sayHelloResponse>
         <result>Hello MyRpcLiteralMessage79</result>
      </ns0:sayHelloResponse>
   </env:Body>
</env:Envelope>
```



style=”document”
use=”literal”

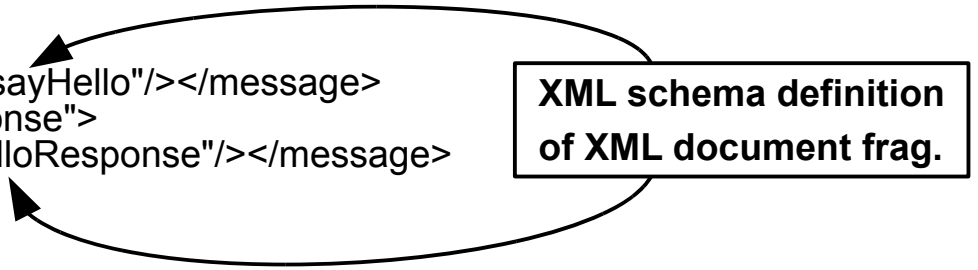
SOAP Binding

document/literal

- SOAP Body element contains an XML document

MyHelloServiceDocLiteral.wsdl(1)

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="urn:Foo"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" name="MyHelloService"
   targetNamespace="urn:Foo">
  <types>
   <schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:soap11-
    enc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" targetNamespace="urn:Foo">
    <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
    <complexType name="sayHello">
     <sequence>
      <element name="String_1" type="string" nillable="true"/>
      <element name="Integer_2" type="int" nillable="true"/>
     </sequence></complexType>
    <complexType name="sayHelloResponse">
     <sequence>
      <element name="result" type="string" nillable="true"/>
     </sequence></complexType>
    <element name="sayHello" type="tns:sayHello"/>
    <element name="sayHelloResponse" type="tns:sayHelloResponse"/>
  </schema></types>
  <message name="HelloIF_sayHello">
   <part name="parameters" element="tns:sayHello"/></message>
  <message name="HelloIF_sayHelloResponse">
   <part name="result" element="tns:sayHelloResponse"/></message>
```



MyHelloServiceDocLiteral.wsdl(2)

```
<portType name="HelloIF">
  <operation name="sayHello">
    <input message="tns:HelloIF_sayHello"/>
    <output message="tns:HelloIF_sayHelloResponse"/></operation></portType>
 <binding name="HelloIFBinding" type="tns:HelloIF">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
     style="document"/>
  <operation name="sayHello">
    <input>
     <soap:body use="literal"/></input>
    <output>
     <soap:body use="literal"/></output>
    <soap:operation soapAction=""/></operation>
  </binding>
 <service name="MyHelloService">
  <port name="HelloIFPort" binding="tns:HelloIFBinding">
    <soap:address xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
     location="http://localhost:8080/hello-jaxrpc/hello"/>
   </port></service></definitions>
```

SOAP Request Message: Doc, Literal

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:tns="urn:Foo"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <tns:sayHello>
      <String_1>MyDocLiteralMessage</String_1>
      <Integer_2>78</Integer_2>
    </tns:sayHello>
  </soap:Body>
</soap:Envelope>
```

```
<complexType name="sayHello">
      <sequence>
        <element name="String_1" type="string" nillable="true"/>
        <element name="Integer_2" type="int"
    nillable="true"/></sequence></complexType>
```

SOAP Response Message: Doc, Literal

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope
    xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:ns0="urn:Foo">
   <env:Body>
      <ns0:sayHelloResponse>
         <result>Hello MyDocLiteralMessage78</result>
      </ns0:sayHelloResponse>
   </env:Body>
</env:Envelope>
```

```
<complexType name="sayHelloResponse">
    <sequence>
      <element name="result" type="string"
    nillable="true"/></sequence></complexType>
```

style=”rpc”
use=”encoded”

SOAP Binding

MyHelloServiceRpcEncoded.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="urn:Foo"
     xmlns:ns2="http://schemas.xmlsoap.org/soap/encoding/"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" name="MyHelloService"
    targetNamespace="urn:Foo">
  <types/>
  <message name="HelloIF_sayHello">
    <part name="String_1" type="xsd:string"/>
    <part name="Integer_2" type="ns2:int"/></message>
  <message name="HelloIF_sayHelloResponse">
    <part name="result" type="xsd:string"/></message>
  <portType name="HelloIF">
    <operation name="sayHello" parameterOrder="String_1 Integer_2">
      <input message="tns:HelloIF_sayHello"/>
      <output message="tns:HelloIF_sayHelloResponse"/></operation></portType>
  <binding name="HelloIFBinding" type="tns:HelloIF">
    <operation name="sayHello">
      <input>
        <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
         namespace="urn:Foo"/></input>
      <output>
        <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
         namespace="urn:Foo"/></output>
      <soap:operation soapAction=""/></operation>
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/></binding>
  <service name="MyHelloService">
    <port name="HelloIFPort" binding="tns:HelloIFBinding">
      <soap:address xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
       location="http://localhost:8080/hello-jaxrpc/hello"/></port></service></definitions>
```

SOAP Request Message: rpc, encoded

```
<<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:n="urn:Foo"
  xmlns:ns2="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <n:sayHello>
      <String_1 xsi:type="xs:string">MyRpcEncodingMessage</String_1>
      <Integer_2 xsi:type="ns2:int">77</Integer_2>
    </n:sayHello>
  </soap:Body>
</soap:Envelope>
```

SOAP Response Message: rpc, encoded

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:ns0="urn:Foo"
   env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
     <env:Body>
        <ns0:sayHelloResponse>
           <result xsi:type="xsd:string">Hello MyRpcEncodingMessage77</result>
        </ns0:sayHelloResponse>
     </env:Body>
</env:Envelope>
```

style=”document”
use=”encoded”

SOAP Binding

SOAP Message Example2: “document” & “encoded”

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:tns="http://tempuri.org/" xmlns:types="http://tempuri.org/encodedTypes"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

  <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

    <types:HelloEncodedWorldResponse
      xsi:type="types:HelloEncodedWorldResponse">
      <HelloEncodedWorldResult href="#id1" />
    </types:HelloEncodedWorldResponse>

    <types:MethodDuration id="id1" xsi:type="types:MethodDuration">
      <start xsi:type="xsd:dateTime">dateTime</start>
      <end xsi:type="xsd:dateTime">dateTime</end>
      <rVal xsi:type="xsd:string">string</rVal>
    </types:MethodDuration>

  </soap:Body>

</soap:Envelope
```

When to Use What?

SOAP Binding

RPC vs. Document-style

RPC

- Procedure call
- Method signature
- Marshaling
- Tightly-coupled
- Point to point
- Synchronous
- Typically within Intranet

Document-style

- Business documents
- Schema
- Parsing & Validating
- Loosely coupled
- End to end
- Asynchronous
- Typically over internet

When to use Which model?

RPC

- Within Enterprise
- Simple, point-to-point
- Short running business process
- Reliable and high bandwidth
- Trusted environment

Document-style

- Between enterprise and enterprise
- Complex, end to end with intermediaries
- Long running business process
- Unpredictable bandwidth
- Blind trust

Thank you!

Sang Shin
Michèle Garoche
www.javapassion.com
"Learning is fun!"