WSDL Binding Styles

Prof. Vipul K. Dabhi
Department of Information Technology,
D. D. University

WSDL Binding Style

- A WSDL document describes a web service.
- A WSDL binding describes how the service is bound to a messaging protocol, particularly the SOAP messaging protocol.
- A WSDL SOAP binding can be either a Remote Procedure Call (RPC) style binding or a document style binding. A SOAP binding can also have an encoded use or a literal use.

WSDL Binding Style

- This gives you four style/use models:
  - RPC/encoded
  - RPC/literal
  - Document/encoded (No Body use)
  - Document/literal
  - Document/literal Wrapped
- Literal means that the SOAP body follows an XML schema, which is included in the web service's WSDL document. As long as the client has access to the WSDL, it knows exactly how each message is formatted.
- Encoded, on the other hand, means that the SOAP body does not follow a schema, but still follows a specific format which the client is expected to already know.

WSDL Binding Style

- public void myMethod(int x, float y);
- Take the method and run it through your favorite Java-to-WSDL tool, specifying that you want it to generate RPC/encoded WSDL.

RPC/encoded WSDL for myMethod

- <message name="myMethodRequest">
  <part name="x" type="xsd:int"/>
  <part name="y" type="xsd:float"/>
  </message>
  <message name="empty"/>

```
<portType name="PT">
   <operation name="myMethod">
      <input message="myMethodRequest"/>
      <output message="empty"/>
   </operation>
</portType>
```

RPC/encoded SOAP message for myMethod

- Now invoke this method with "5" as the value for parameter x and "5.0" for parameter y.
- <soap:envelope>
- <soap:body>
- <myMethod>
- <x xsi:type="xsd:int">5</x>
- <y xsi:type="xsd:float">5.0</y>
- </myMethod>
- </soap:body>
- </soap:envelope>

Strength / Weakness of RPC Encoded Style

- Strengths
  - The WSDL is about as straightforward as it's possible for WSDL to be.
  - The operation name appears in the message, so the receiver has an easy time dispatching this message to the implementation of the operation.

- Weakness
  - The type encoding info (such as xsi:type="xsd:int") is usually just overhead which degrades throughput performance.
  - You cannot easily validate this message since only the <x ...>5</x> and <y ...>5.0</y> lines contain things defined in a schema; the rest of the soap:body contents comes from WSDL definitions.

RPC/literal WSDL for myMethod

```
<message name="myMethodRequest">
  <part name="x" type="xsd:int"/>
  <part name="y" type="xsd:float"/>
</message>
<message name="empty"/>

<portType name="PT">
  <operation name="myMethod">
    <input message="myMethodRequest"/>
    <output message="empty"/>
  </operation>
</portType>
```

```
<types>
  <schema>
    <element name="xElement" type="xsd:int"/>
    <element name="yElement" type="xsd:float"/>
  </schema>
</types>

<message name="myMethodRequest">
  <part name="x" element="xElement"/>
  <part name="y" element="yElement"/>
</message>
<message name="empty"/>

<portType name="PT">
  <operation name="myMethod">
    <input message="myMethodRequest"/>
    <output message="empty"/>
  </operation>
</portType>
```

RPC/literal SOAP message for myMethod

- <soap:envelope>
- <soap:body>
- <myMethod>
- <x>5</x>
- <y>5.0</y>
- </myMethod>
- </soap:body>
- </soap:envelope>

RPC/literal

- Strengths
  - The WSDL is still about as straightforward as it is possible for WSDL to be.
  - The operation name still appears in the message.
  - The type encoding info is eliminated.
- Weakness
  - You still cannot easily validate this message since only the <x ...>5</x> and <y ...>5.0</y> lines contain things defined in a schema;

Document/literal WSDL for myMethod

```
<types>
   <schema>
      <element name="xElement" type="xsd:int"/>
      <element name="yElement" type="xsd:float"/>
   </schema>
</types>

<message name="myMethodRequest">
   <part name="x" element="xElement"/>
   <part name="y" element="yElement"/>
</message>
<message name="empty"/>

<portType name="PT">
   <operation name="myMethod">
      <input message="myMethodRequest"/>
      <output message="empty"/>
   </operation>
</portType>
```

Document/literal SOAP message for myMethod

- <soap:envelope>
- <soap:body>
- <xElement>5</xElement>
- <yElement>5.0</yElement>
- </soap:body>
- </soap:envelope>

Strength/Weakness of Document/literal

- Strengths
  - There is no type encoding info.
  - You can finally validate this message with any XML validator. Everything within the soap:body is defined in a schema.
- Weakness
  - The WSDL is getting a bit more complicated
  - The operation name in the SOAP message is lost. Without the name, dispatching can be difficult, and sometimes impossible

Document/literal wrapped WSDL for myMethod

```
<types>
  <schema>
    <element name="myMethod">
      <complexType>
        <sequence>
          <element name="x" type="xsd:int"/>
          <element name="y" type="xsd:float"/>
        </sequence>
      </complexType>
    </element>
    <element name="myMethodResponse">
      <complexType/>
    </element>
  </schema>
</types>
<message name="myMethodRequest">
  <part name="parameters" element="myMethod"/>
</message>
<message name="empty">
  <part name="parameters" element="myMethodResponse"/>
</message>

<portType name="PT">
  <operation name="myMethod">
    <input message="myMethodRequest"/>
    <output message="empty"/>
  </operation>
</portType>
```

Document/literal wrapped SOAP message for myMethod

```
<soap:envelope>
    <soap:body>
        <myMethod>
            <x>5</x>
            <y>5.0</y>
        </myMethod>
    </soap:body>
</soap:envelope>
```

Document/literal

- Strength
  - There is no type encoding info.
  - Everything that appears in the soap:body is defined by the schema, so you can easily validate this message.
  - Once again, you have the method name in the SOAP message.
- Weakness
  - The WSDL is even more complicated.

RPC Vs Document

- The body of an RPC (remote procedure call) style SOAP message is constructed in a specific way, which is defined in the SOAP standard.
- The message body contains an XML element for each "parameter" of the method.
- These parameter elements are wrapped in an XML element which contains the name of the method that is being called.
- The WSDL code for a RPC-style web service is less complex than that of a document-style web service.

RPC Vs Document

- A document style web service, on the other hand, contains no restrictions for how the SOAP body must be constructed.
- It allows you to include whatever XML data you want and also to include a schema for this XML.
- This means that the client's and server's application code must do the marshalling and unmarshalling work
- The WSDL code for a document-style web service is much more complex than that of a RPC-style web service