**Binding Style**

Let’s see the different Binding style. ☺

As we have already discussed that Soap messages (Soap envelope) are exchanged between Service Consumer and Server.

How this Soap message is formed? confused-smiley

Soap messages are formed from WSDL☺. The WSDL element **binding** basically tells how to **translate a** WSDL binding to a **SOAP message**.

* A WSDL binding describes how the service is bound to a messaging protocol like (http, https etc.)

There are basically two types of **SOAP** **binding**.

* RPC
* Document

**RPC** (Remote procedure call)

**Document**

**Soap** **binding**

A SOAP binding can also **use** **encoded** or **literal** style.

The terms **encoded** and **literal** are only meaningful for the **WSDL-to-SOAP mapping.**

That gives four different combinations.

* RPC/encoded
* RPC/literal
* Document/encoded
* Document/literal

Let’s see all by example.

Suppose we have **Calculator** service that provides **sum** method as a service.

**See method:-**

**public** **float** sum(**float** firstNumber, **int** secondNumber );

**WSDL Binding :- Looks like**

**<wsdl:binding name=*"CalculatorServiceSoapBinding"* type=*"tns:ICalculator"*>**

**<soap:binding style=*"RPC"* transport=*"http://schemas.xmlsoap.org/soap/http"*/>**

**<wsdl:operation name=*"sum"*>**

**<soap:operation soapAction=*"urn:Sum"* style=*"RPC"*/>**

**<wsdl:input name=*"sum"*>**

**<soap:body use=*"encoded"*/>**

**</wsdl:input>**

**<wsdl:output name=*"sumResponse"*>**

**<soap:body use=*"encoded"*/>**

**</wsdl:output>**

**</wsdl:operation>**

**RPC/encoded :-**

**<soap:envelope>**

**<soap:body>**

**<sum>**

**<firstNumbe > 7.5</ firstNumbe>**

**<** **secondNumber >2</** **secondNumber** **>**

**</sum>**

**</soap:body>**

**</soap:envelope>**

Here we can validate only child tag of <sum> against schema res

Same as Document/literal only difference is method name is wrapped.

Now it looks like **RPC/literal** but actually in **RPC/literal** that is method/operation name e.g. here in RPC/literal **<sum>** is method name.

In **RPC/literal** **wrapped** e.g. **<sum>** it is not **method/operation** name. It is **wrapped element**.

**portType** one the tag in **WSDL**  contain the **input element** tag.

One of the characteristics of the **wrapped pattern** is that the name of the **input element** is the same as the name of the **operation/method**.

When we use **RPC/literal** **wrapped** it wrap the arguments by taking the name of **input element** from **portType** tag.

**RPC/literal** **wrapped** is **WS-I compliant** because there **is no type information** and soap body **has only one child**.

**<wsdl:portType name=*"ICalculator"*>**

**<wsdl:operation name=*"sum"*>**

**<wsdl:input name=*"sum"* message=*"tns:sum"*>**

**</wsdl:input>**

**<wsdl:output name=*"sumResponse"* message=*"tns:sumResponse"*>**

**</wsdl:output>**

**</wsdl:operation>**

**</wsdl:portType>**

As per our requirement we can chose the binding style.

*Binding style helps at server side to dispatch on the operation/method.*

**Document / literal wrapped**

**Public void sum( int x, int y );**

**Public void sum( int x, int y, float z );**

**Don’t** use in the case of **function overloading**. Because in wrapped pattern it try to generate two elements with the same name that is **WSDL** **generation** error.

***Tips: - SO avoid it in case of function overloading.***

**Document / literal**

**This binding is good in case of function overloading.**

**Public void sum( int x, int y );**

**Public void sum( int x, int y, float z );**

This binding gets fail if two methods having **different name** but **same number of arguments**. E.g.

Public void **sum**( int x, int y, float z );

Public void **mul**( int x, int y, float z );

***Tips: - SO avoid it in case if different function has same no. of arguments.***

***RPC / literal***

*This binding is good in case of* ***function overloading*** *and also good if two* ***different methods have same no. of arguments.***

Public void sum( int x, int y);

Public void sum( int x, int y, float z );

Public void mul( int x, int y, float z );