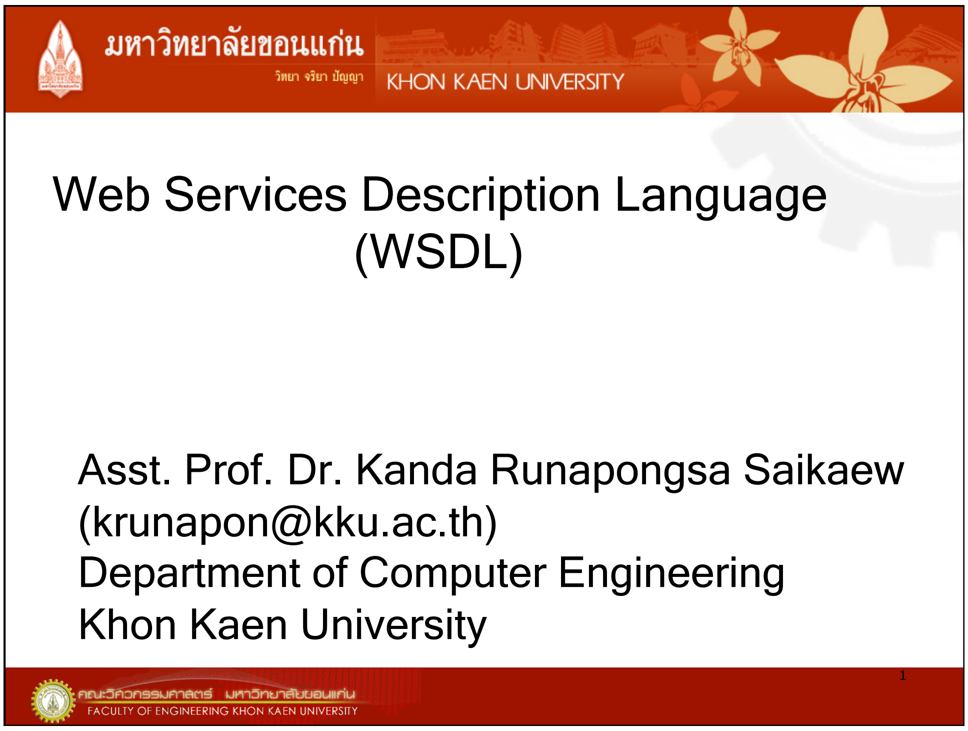


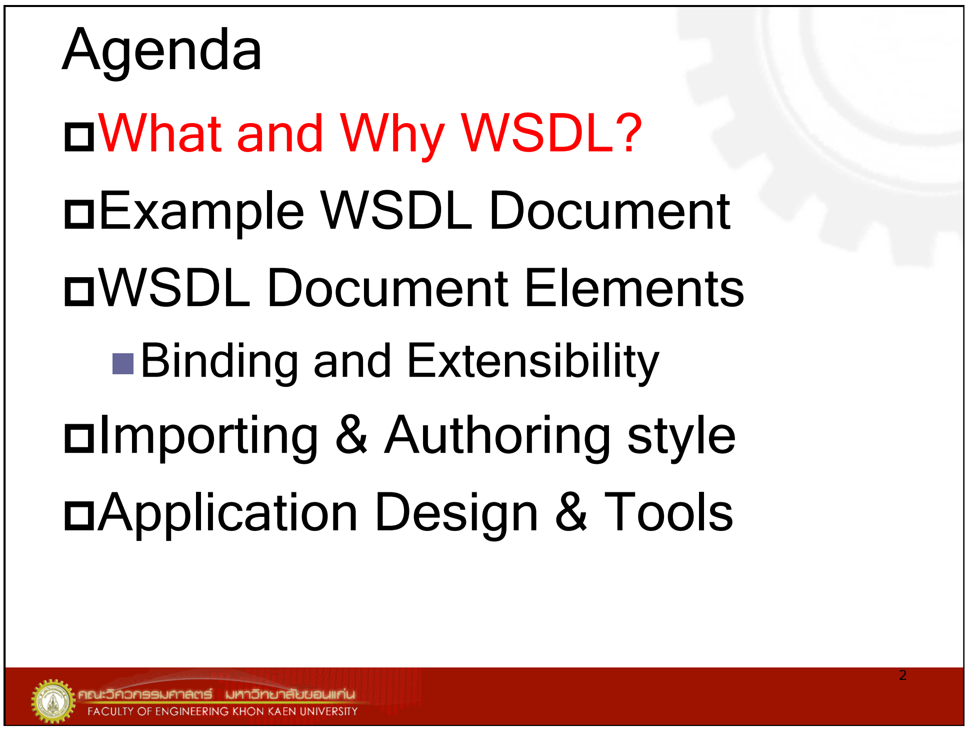
มหาวิทยาลัยขอนแก่น
วิทยา จริยา ปัญญา
KHON KAEN UNIVERSITY

Web Services Description Language (WSDL)

Asst. Prof. Dr. Kanda Runapongsa Saikaew
(krunapon@kku.ac.th)
Department of Computer Engineering
Khon Kaen University

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY

1

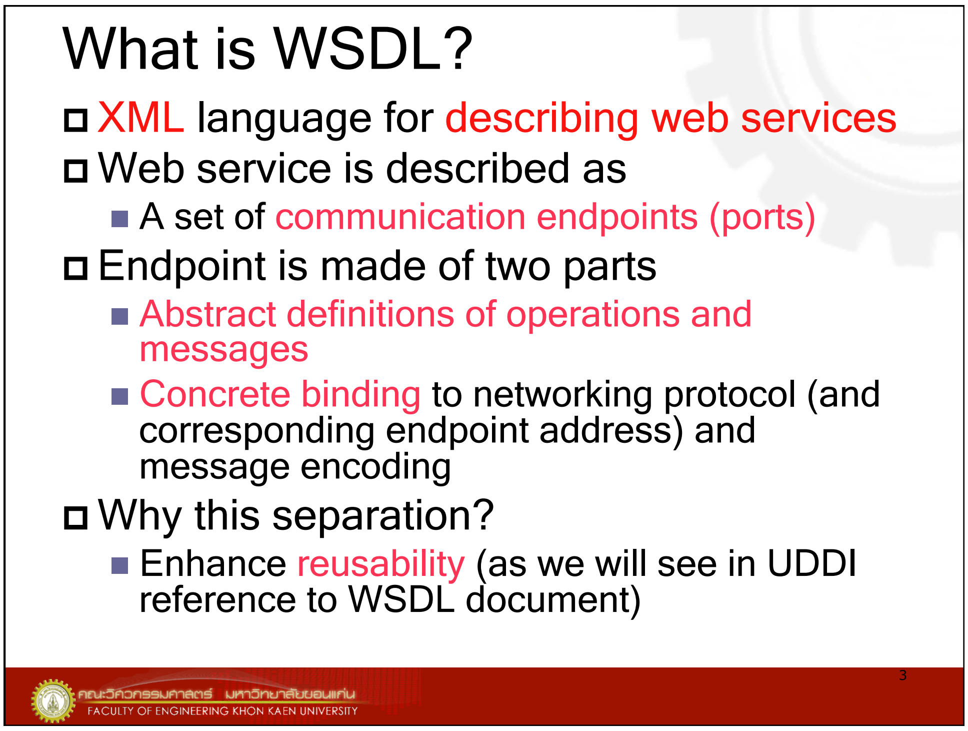


Agenda

- What and Why WSDL?
- Example WSDL Document
- WSDL Document Elements
  - Binding and Extensibility
- Importing & Authoring style
- Application Design & Tools

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY

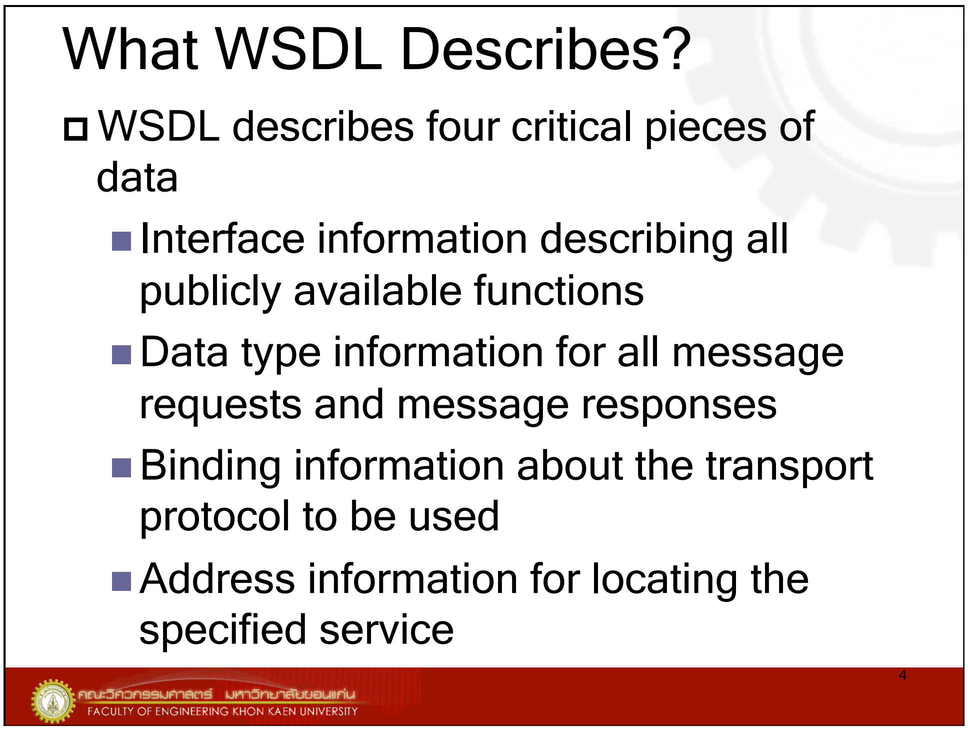
2



What is WSDL?

- XML language for describing web services
- Web service is described as
  - A set of communication endpoints (ports)
- Endpoint is made of two parts
  - Abstract definitions of operations and messages
  - Concrete binding to networking protocol (and corresponding endpoint address) and message encoding
- Why this separation?
  - Enhance reusability (as we will see in UDDI reference to WSDL document)

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 3

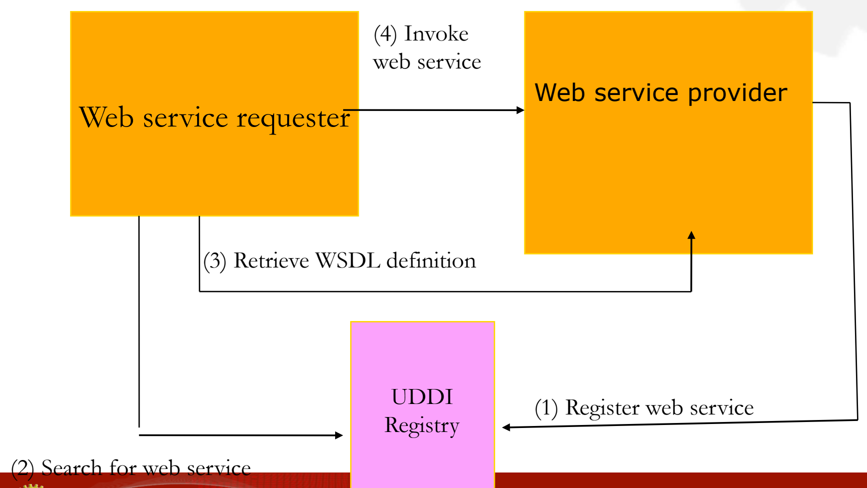


What WSDL Describes?

- WSDL describes four critical pieces of data
  - Interface information describing all publicly available functions
  - Data type information for all message requests and message responses
  - Binding information about the transport protocol to be used
  - Address information for locating the specified service

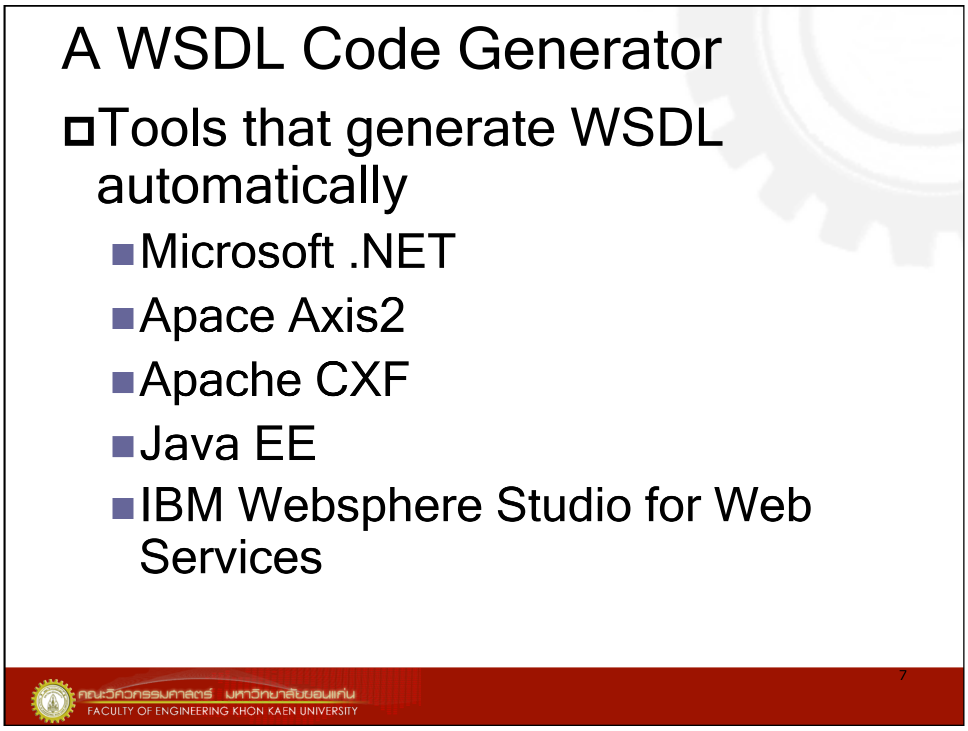
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 4

Where is WSDL Used?



Why WSDL?

- Enables automation of communication details between communicating partners
  - Machines can read WSDL
  - Machines can invoke a service defined in WSDL
- Discoverable through registry
- Arbitration
  - 3rd party can verify if communication conforms to WSDL

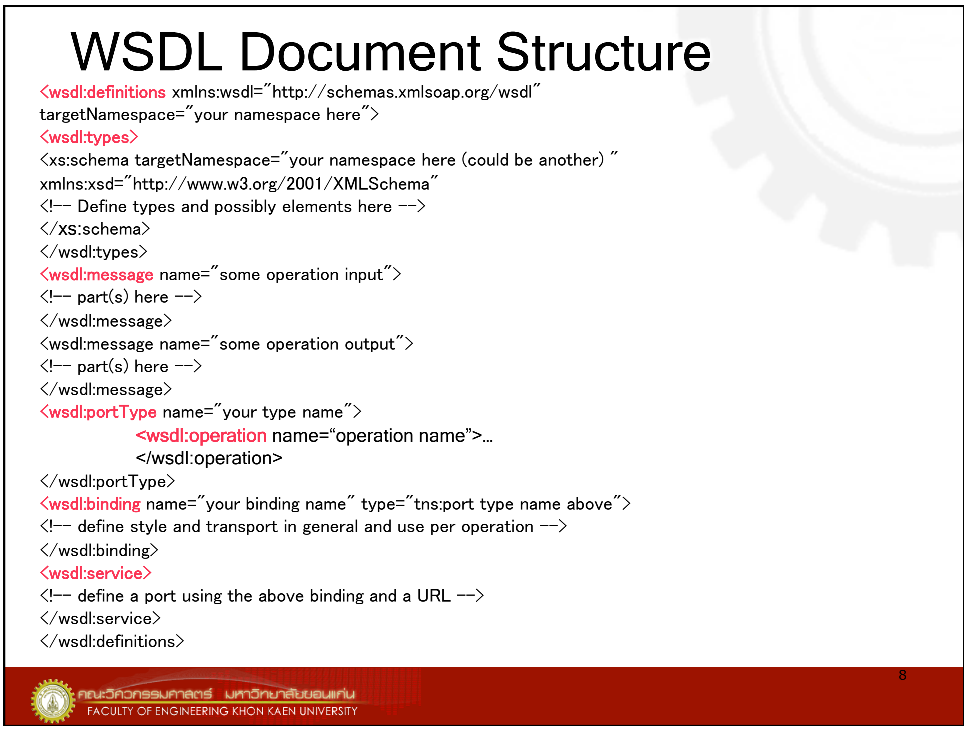


A WSDL Code Generator

- Tools that generate WSDL automatically
  - Microsoft .NET
  - Apace Axis2
  - Apache CXF
  - Java EE
  - IBM Websphere Studio for Web Services

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY

7

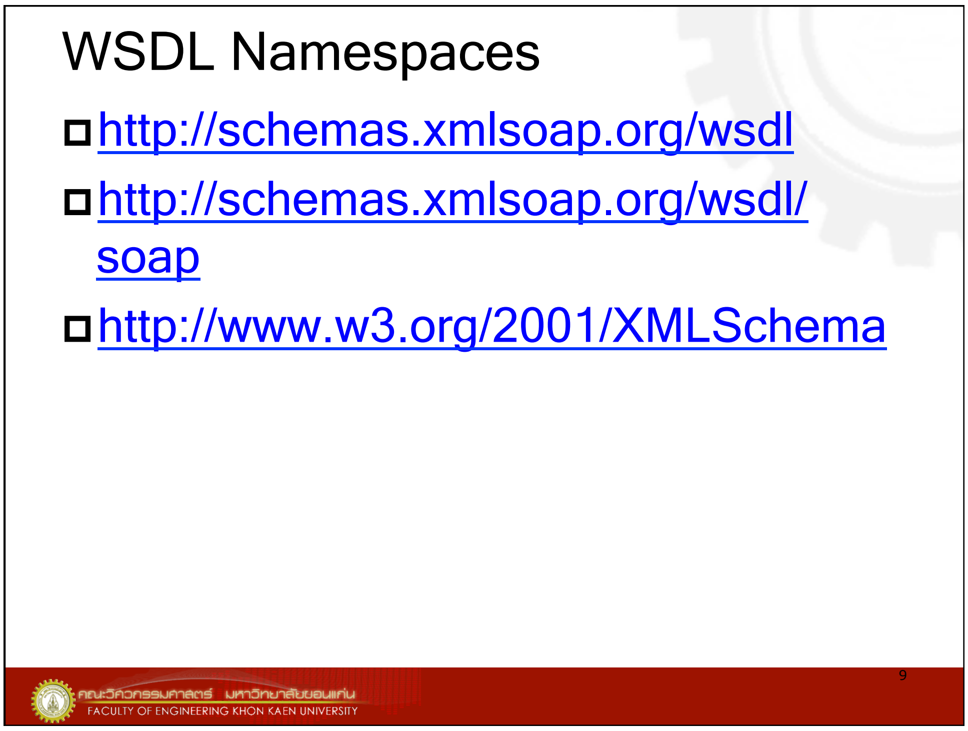


WSDL Document Structure

```
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl"
targetNamespace="your namespace here">
<wsdl:types>
<xs:schema targetNamespace="your namespace here (could be another) "
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
<!-- Define types and possibly elements here -->
</xs:schema>
</wsdl:types>
<wsdl:message name="some operation input">
<!-- part(s) here -->
</wsdl:message>
<wsdl:message name="some operation output">
<!-- part(s) here -->
</wsdl:message>
<wsdl:portType name="your type name">
        <wsdl:operation name="operation name">…
        </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="your binding name" type="tns:port type name above">
<!-- define style and transport in general and use per operation -->
</wsdl:binding>
<wsdl:service>
<!-- define a port using the above binding and a URL -->
</wsdl:service>
</wsdl:definitions>
```

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY

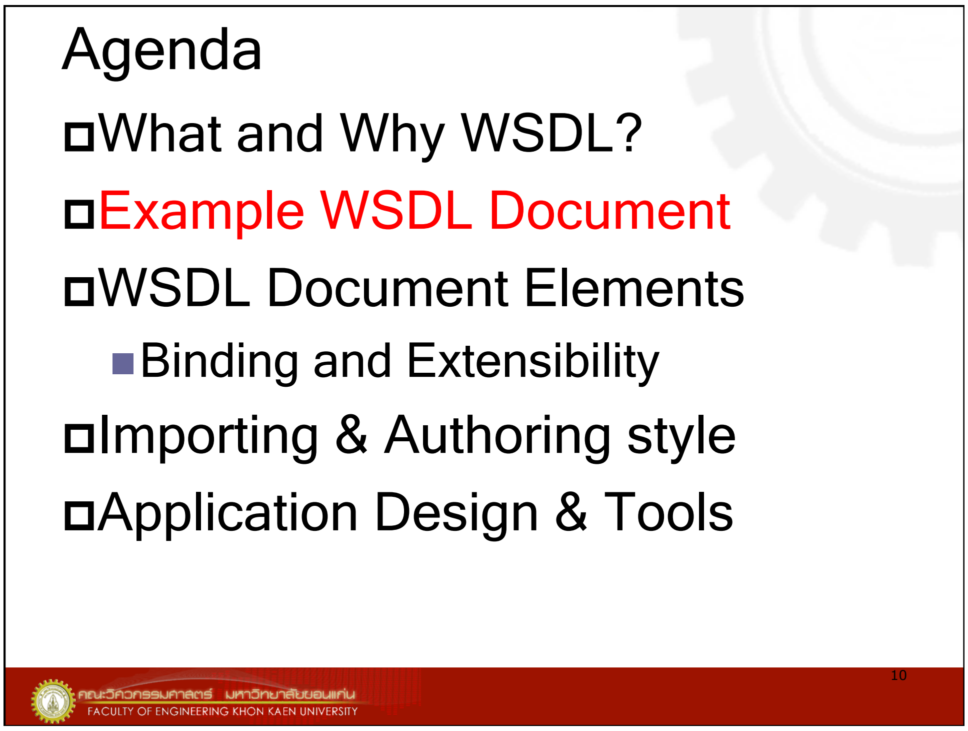
8



WSDL Namespaces

- http://schemas.xmlsoap.org/wsdl
- http://schemas.xmlsoap.org/wsdl/soap
- http://www.w3.org/2001/XMLSchema

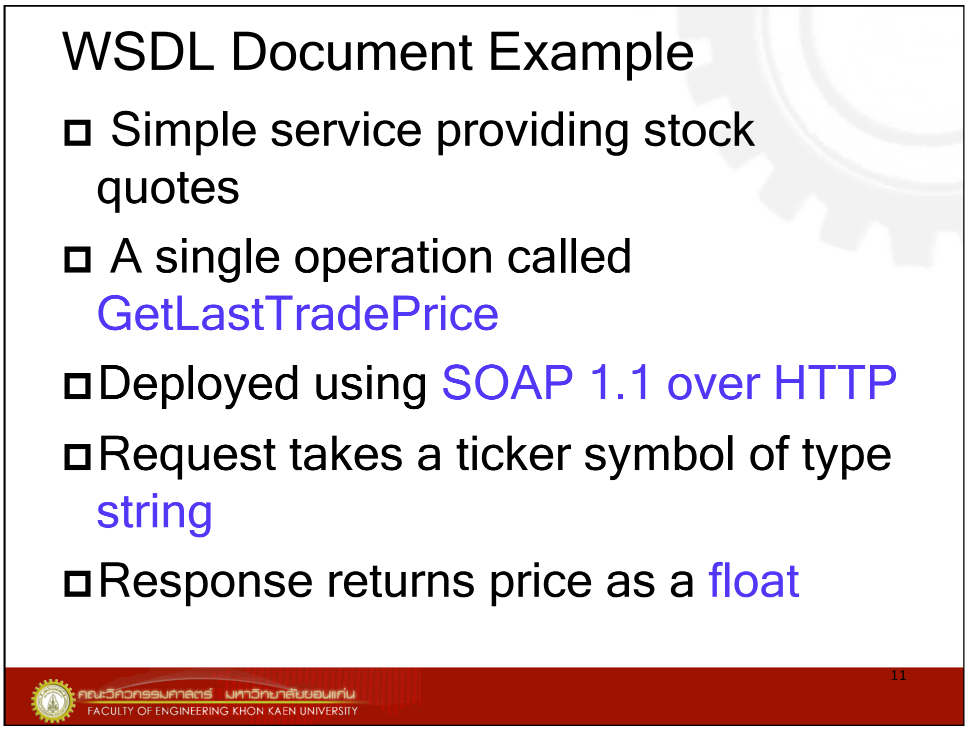
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY
9



Agenda

- What and Why WSDL?
- Example WSDL Document
- WSDL Document Elements
  - Binding and Extensibility
- Importing & Authoring style
- Application Design & Tools

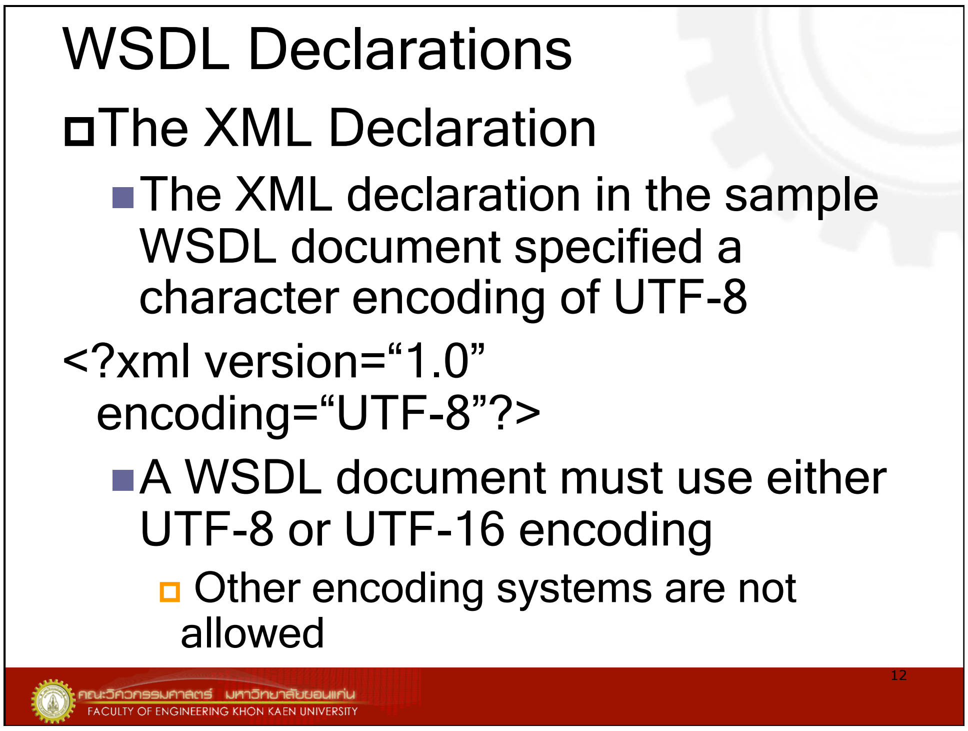
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY
10



WSDL Document Example

- Simple service providing stock quotes
- A single operation called GetLastTradePrice
- Deployed using SOAP 1.1 over HTTP
- Request takes a ticker symbol of type string
- Response returns price as a float

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY
11



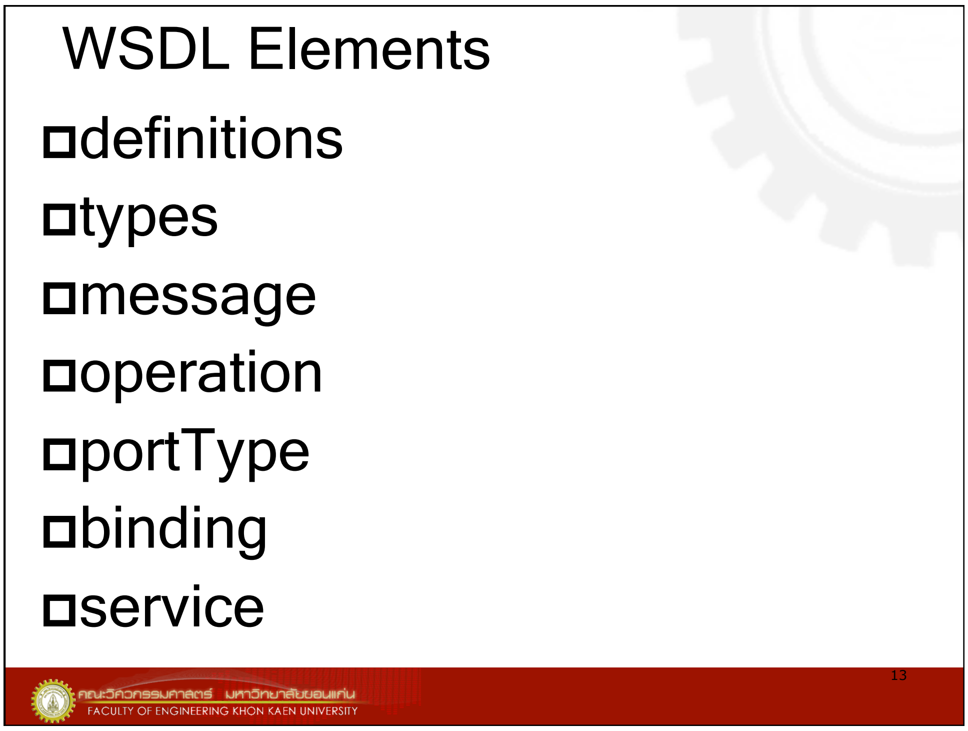
WSDL Declarations

- The XML Declaration
  - The XML declaration in the sample WSDL document specified a character encoding of UTF-8

<?xml version="1.0" encoding="UTF-8"?>

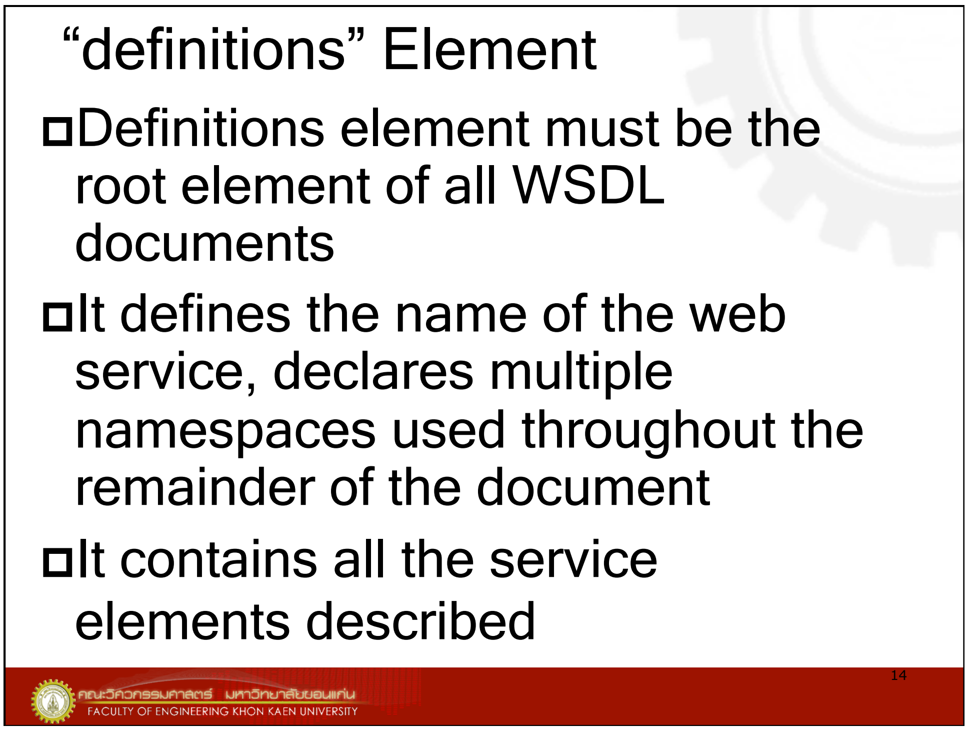
  - A WSDL document must use either UTF-8 or UTF-16 encoding
    - Other encoding systems are not allowed

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY
12



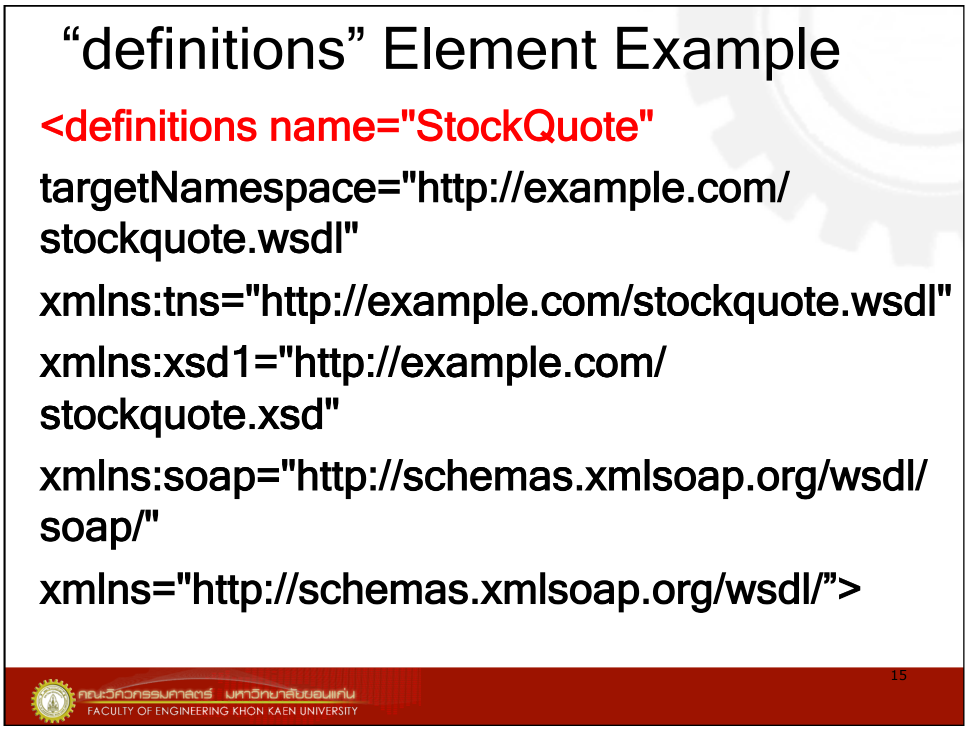
WSDL Elements

- definitions
- types
- message
- operation
- portType
- binding
- service



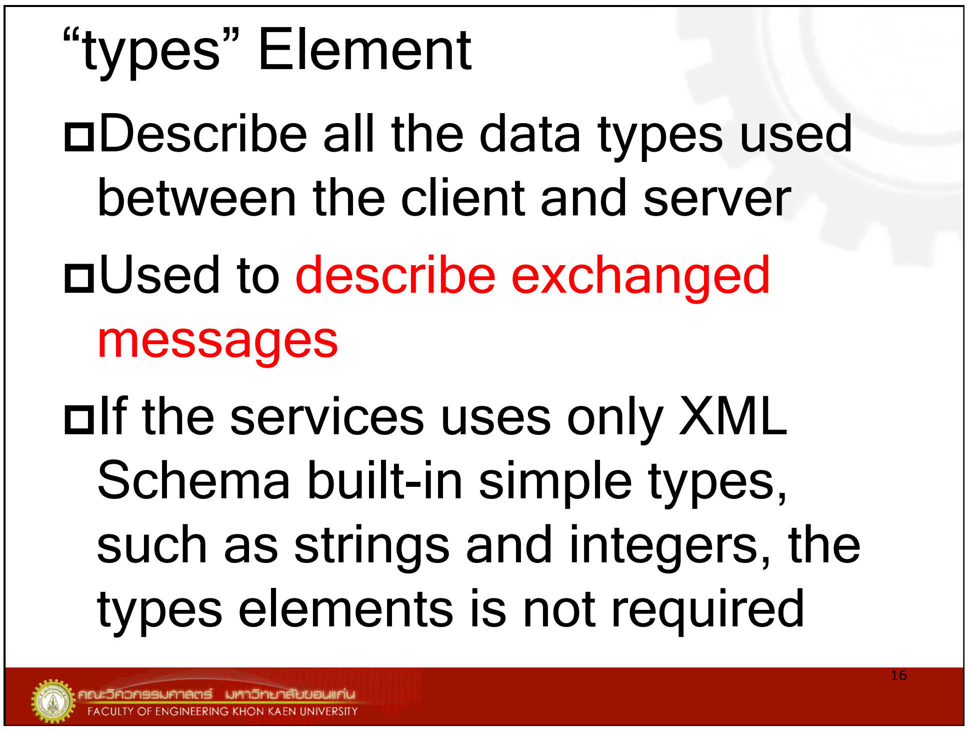
“definitions” Element

- Definitions element must be the root element of all WSDL documents
- It defines the name of the web service, declares multiple namespaces used throughout the remainder of the document
- It contains all the service elements described



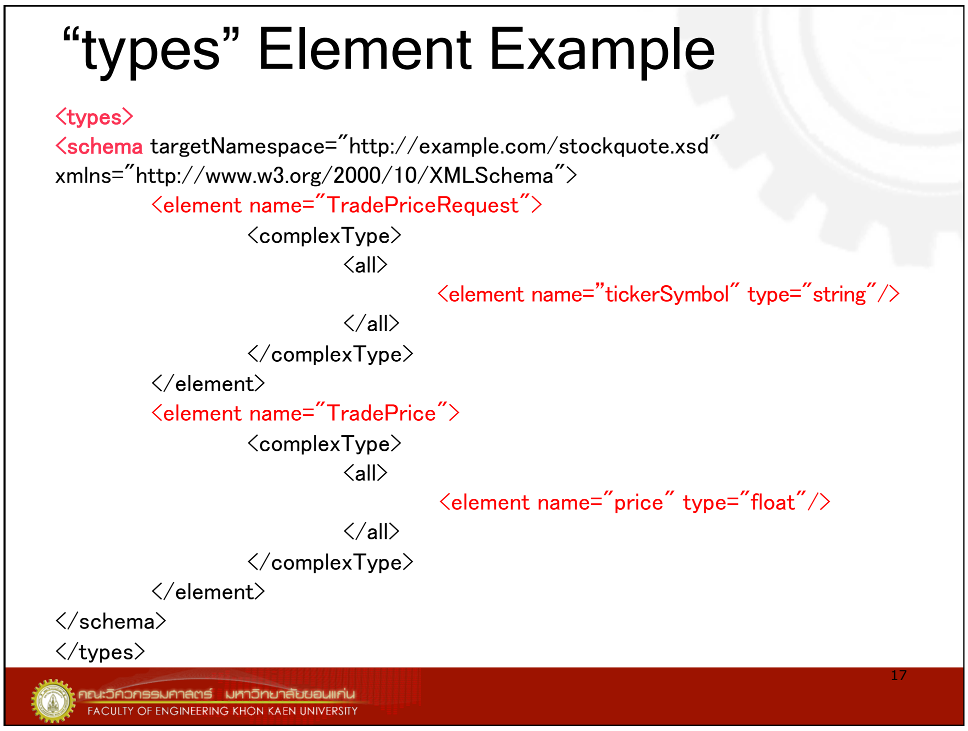
“definitions” Element Example

```
<definitions name="StockQuote"
targetNamespace="http://example.com/
stockquote.wsdl"
xmlns:tns="http://example.com/stockquote.wsdl"
xmlns:xsd1="http://example.com/
stockquote.xsd"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/
soap/"
xmlns="http://schemas.xmlsoap.org/wsdl/”>
```



“types” Element

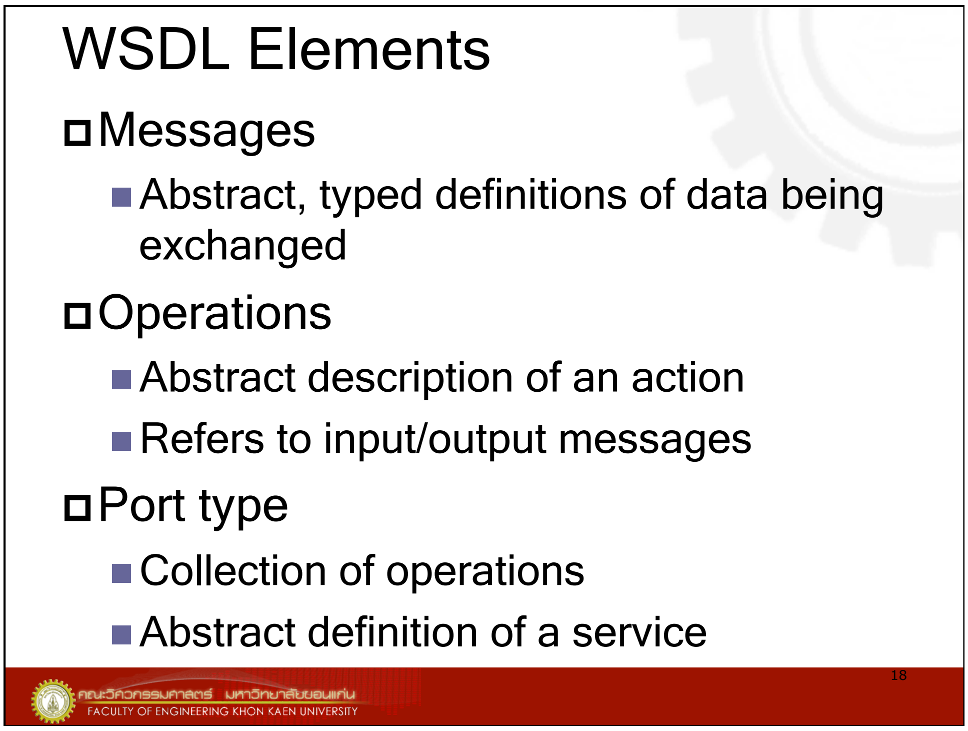
- Describe all the data types used between the client and server
- Used to describe exchanged messages
- If the services uses only XML Schema built-in simple types, such as strings and integers, the types elements is not required



“types” Element Example

```
<types>
<schema targetNamespace="http://example.com/stockquote.xsd"
xmlns="http://www.w3.org/2000/10/XMLSchema">
        <element name="TradePriceRequest">
                <complexType>
                        <all>
                                <element name="tickerSymbol" type="string"/>
                        </all>
                </complexType>
        </element>
        <element name="TradePrice">
                <complexType>
                        <all>
                                <element name="price" type="float"/>
                        </all>
                </complexType>
        </element>
</schema>
</types>
```

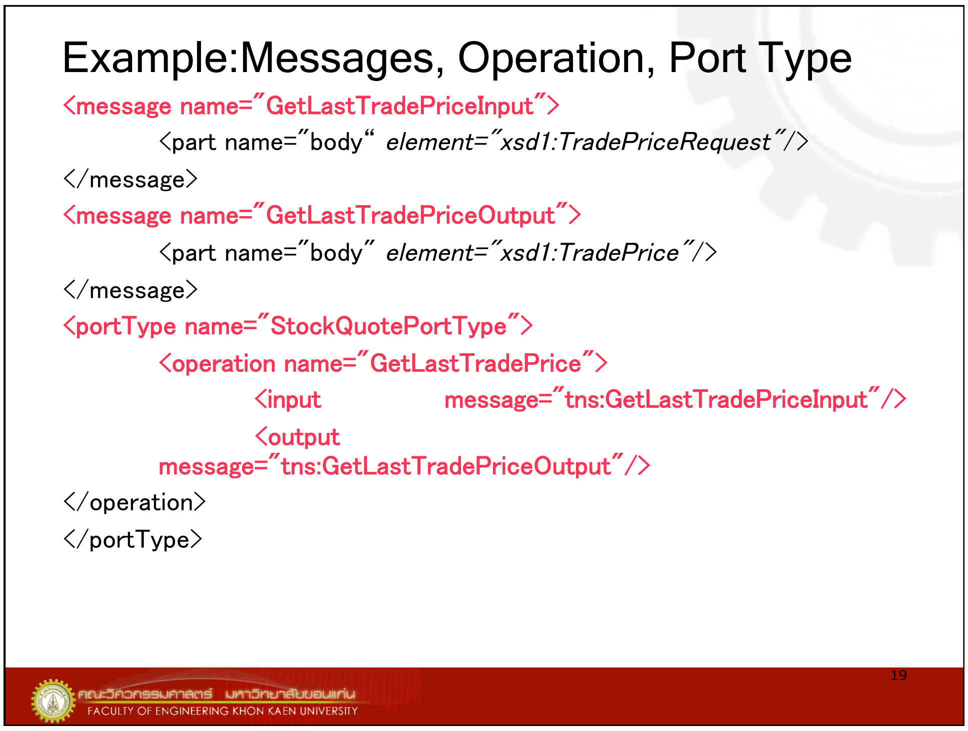
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY
17



WSDL Elements

- Messages
  - Abstract, typed definitions of data being exchanged
- Operations
  - Abstract description of an action
  - Refers to input/output messages
- Port type
  - Collection of operations
  - Abstract definition of a service

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY
18

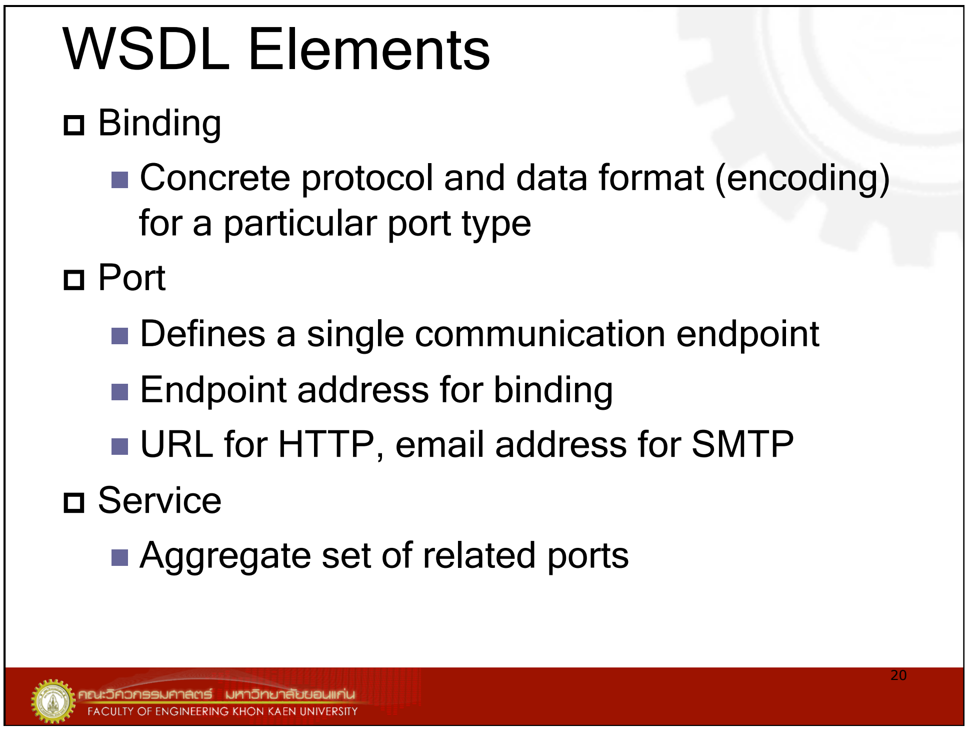


Example:Messages, Operation, Port Type

```
<message name=″GetLastTradePriceInput″>
        <part name=″body“ element=″xsd1:TradePriceRequest″/>
</message>
<message name=″GetLastTradePriceOutput″>
        <part name=″body″ element=″xsd1:TradePrice″/>
</message>
<portType name=″StockQuotePortType″>
        <operation name=″GetLastTradePrice″>
                <input          message=″tns:GetLastTradePriceInput″/>
                <output
        message=″tns:GetLastTradePriceOutput″/>
</operation>
</portType>
```

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY

19

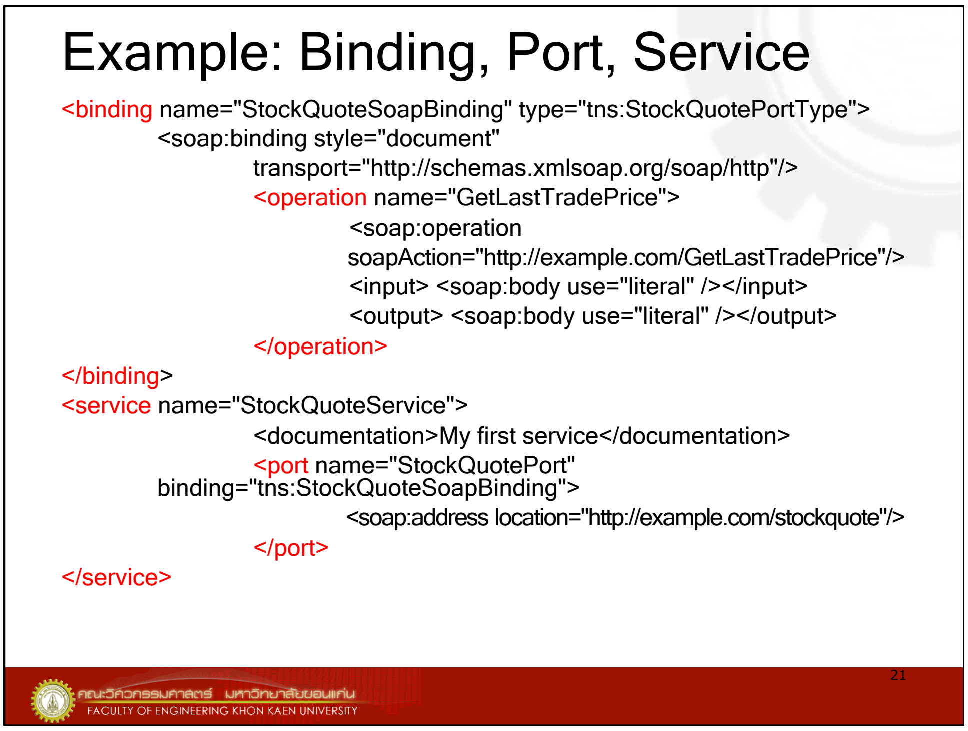


WSDL Elements

- Binding
  - Concrete protocol and data format (encoding) for a particular port type
- Port
  - Defines a single communication endpoint
  - Endpoint address for binding
  - URL for HTTP, email address for SMTP
- Service
  - Aggregate set of related ports

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY

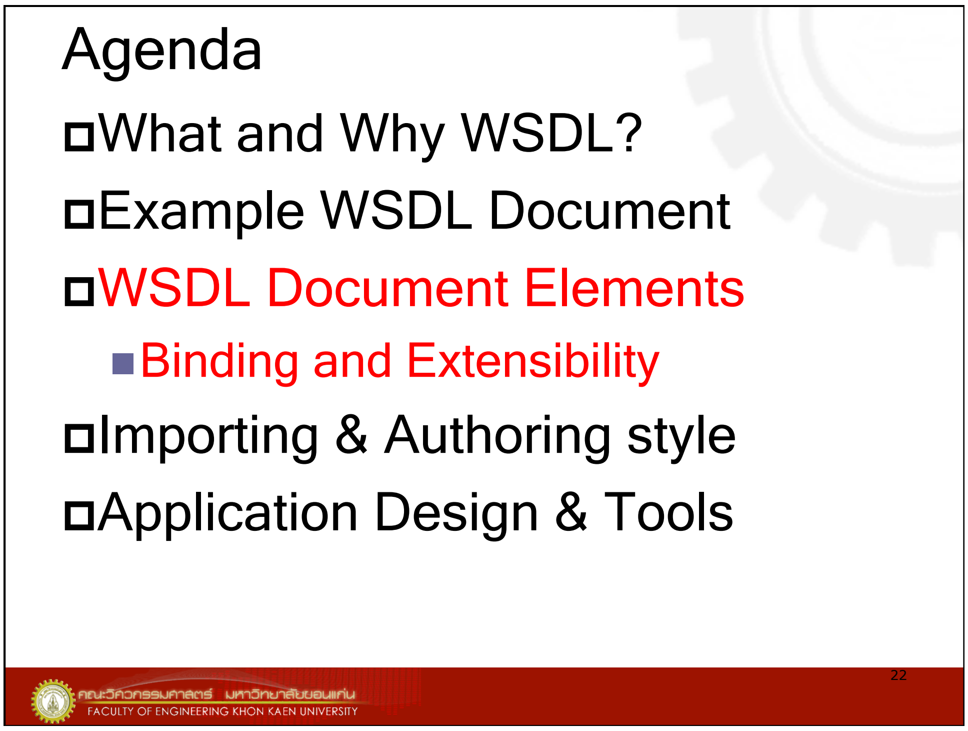
20



Example: Binding, Port, Service

```
<binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
          <soap:binding style="document"
                    transport="http://schemas.xmlsoap.org/soap/http"/>
                    <operation name="GetLastTradePrice">
                              <soap:operation
                              soapAction="http://example.com/GetLastTradePrice"/>
                              <input> <soap:body use="literal" /></input>
                              <output> <soap:body use="literal" /></output>
                    </operation>
</binding>
<service name="StockQuoteService">
                    <documentation>My first service</documentation>
                    <port name="StockQuotePort"
          binding="tns:StockQuoteSoapBinding">
                              <soap:address location="http://example.com/stockquote"/>
                    </port>
</service>
```

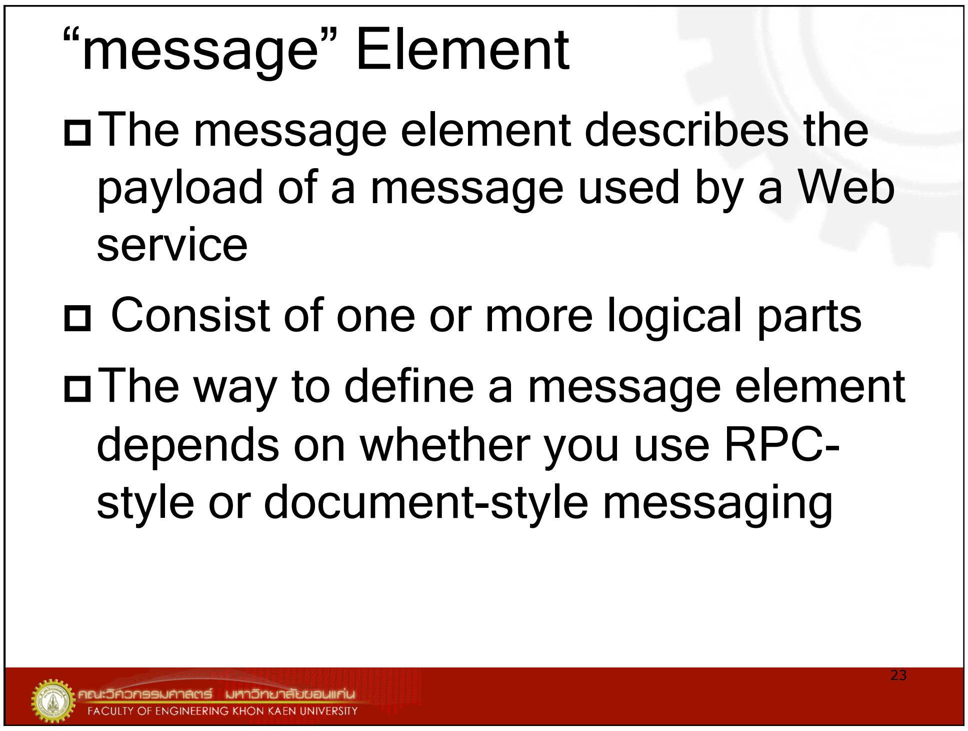
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 21



Agenda

- What and Why WSDL?
- Example WSDL Document
- WSDL Document Elements
  - Binding and Extensibility
- Importing & Authoring style
- Application Design & Tools

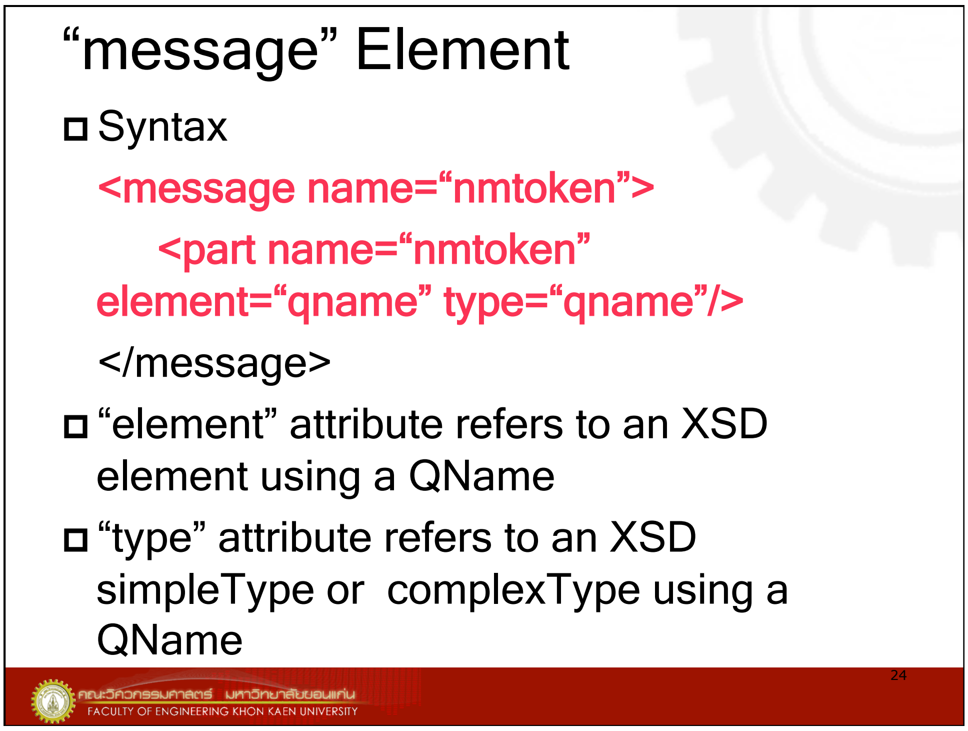
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 22



“message” Element

- The message element describes the payload of a message used by a Web service
- Consist of one or more logical parts
- The way to define a message element depends on whether you use RPC-style or document-style messaging

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 23



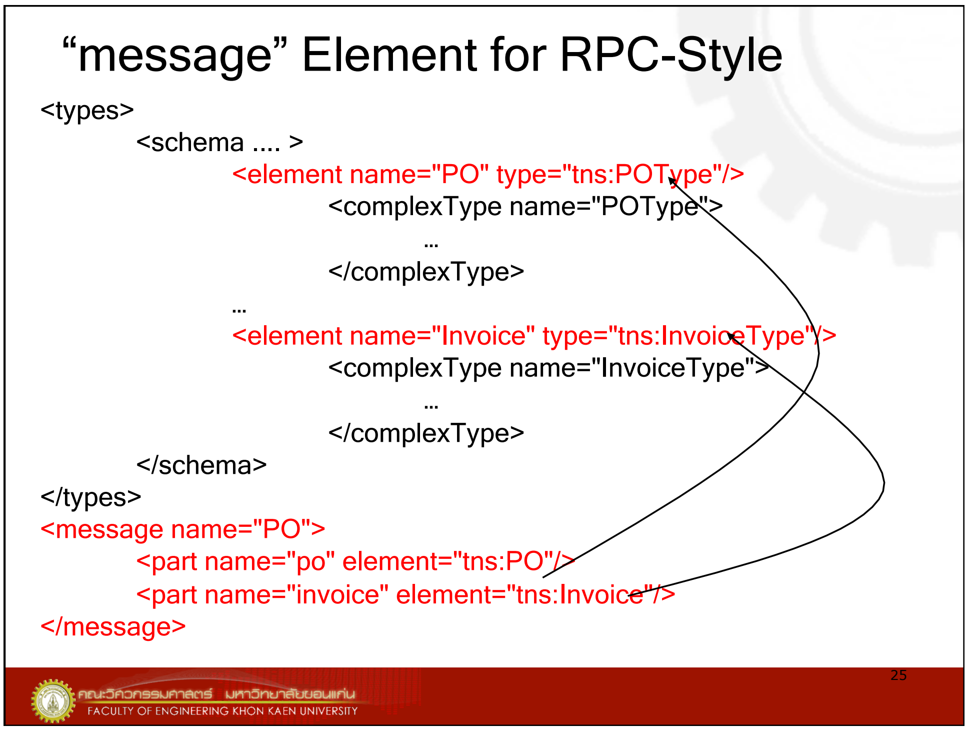
“message” Element

- Syntax

```
<message name=“nmtoken”>
     <part name=“nmtoken”
element=“qname” type=“qname”/>
</message>
```

- “element” attribute refers to an XSD element using a QName
- “type” attribute refers to an XSD simpleType or complexType using a QName

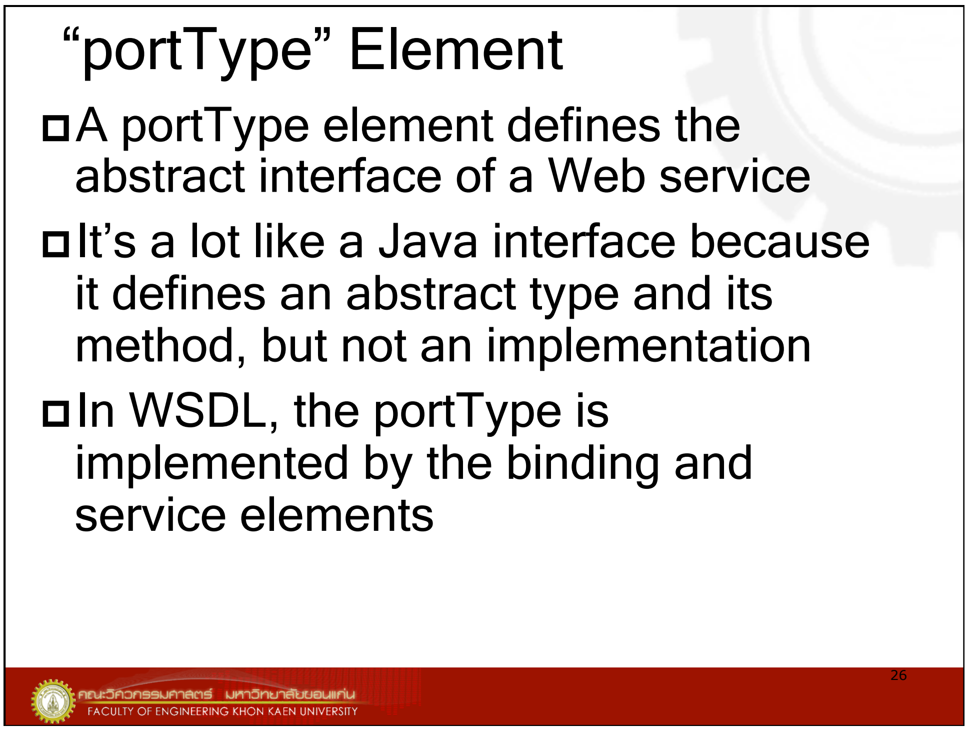
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 24



“message” Element for RPC-Style

```
<types>
        <schema .... >
                <element name="PO" type="tns:POType"/>
                        <complexType name="POType">
                                …
                        </complexType>
                …
                <element name="Invoice" type="tns:InvoiceType"/>
                        <complexType name="InvoiceType">
                                …
                        </complexType>
        </schema>
</types>
<message name="PO">
        <part name="po" element="tns:PO"/>
        <part name="invoice" element="tns:Invoice"/>
</message>
```

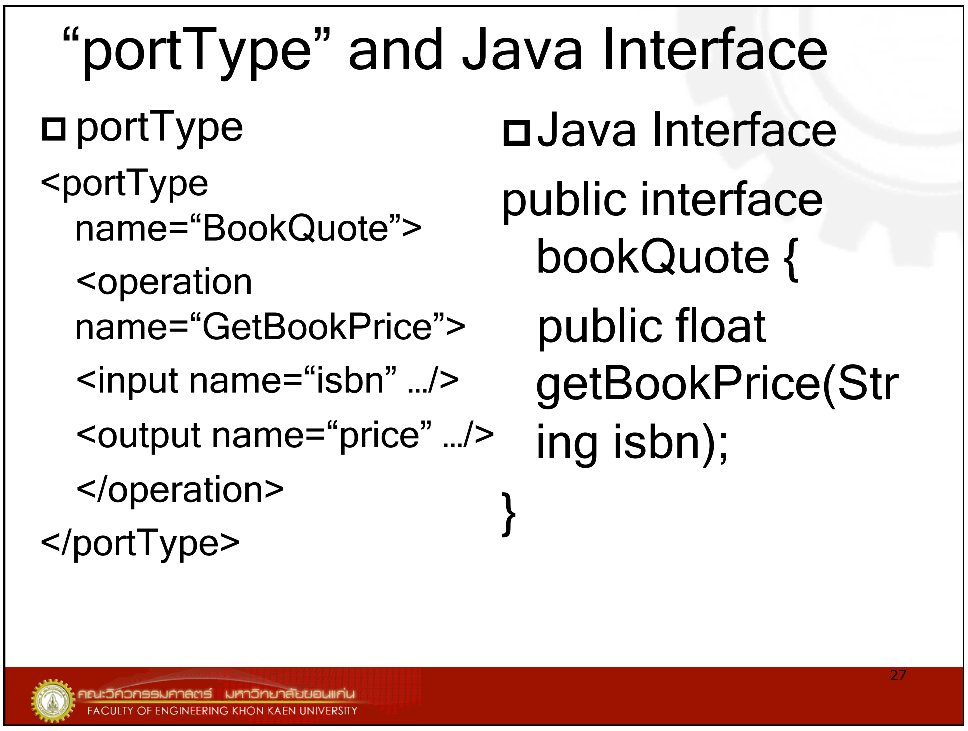
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 25



“portType” Element

- A portType element defines the abstract interface of a Web service
- It’s a lot like a Java interface because it defines an abstract type and its method, but not an implementation
- In WSDL, the portType is implemented by the binding and service elements

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 26



“portType” and Java Interface

- portType

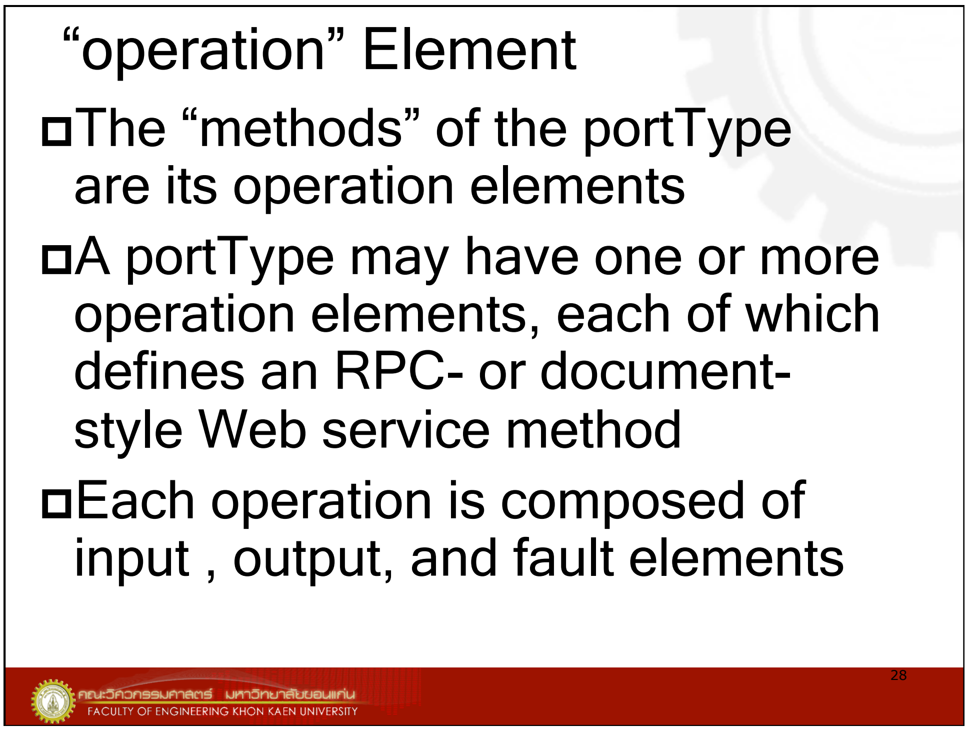
```
<portType
   name=“BookQuote”>
   <operation
   name=“GetBookPrice”>
   <input name=“isbn” …/>
   <output name=“price” …/>
   </operation>
</portType>
```

- Java Interface

```
public interface
   bookQuote {
   public float
   getBookPrice(Str
   ing isbn);
}
```

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY

27

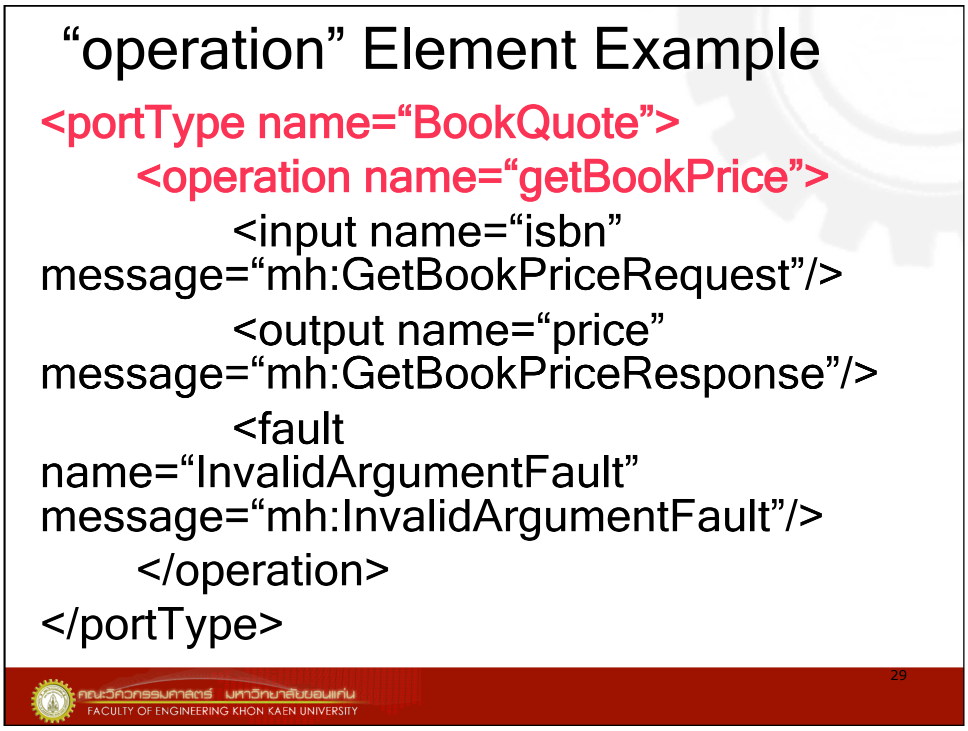


“operation” Element

- The “methods” of the portType are its operation elements
- A portType may have one or more operation elements, each of which defines an RPC- or document-style Web service method
- Each operation is composed of input , output, and fault elements

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY

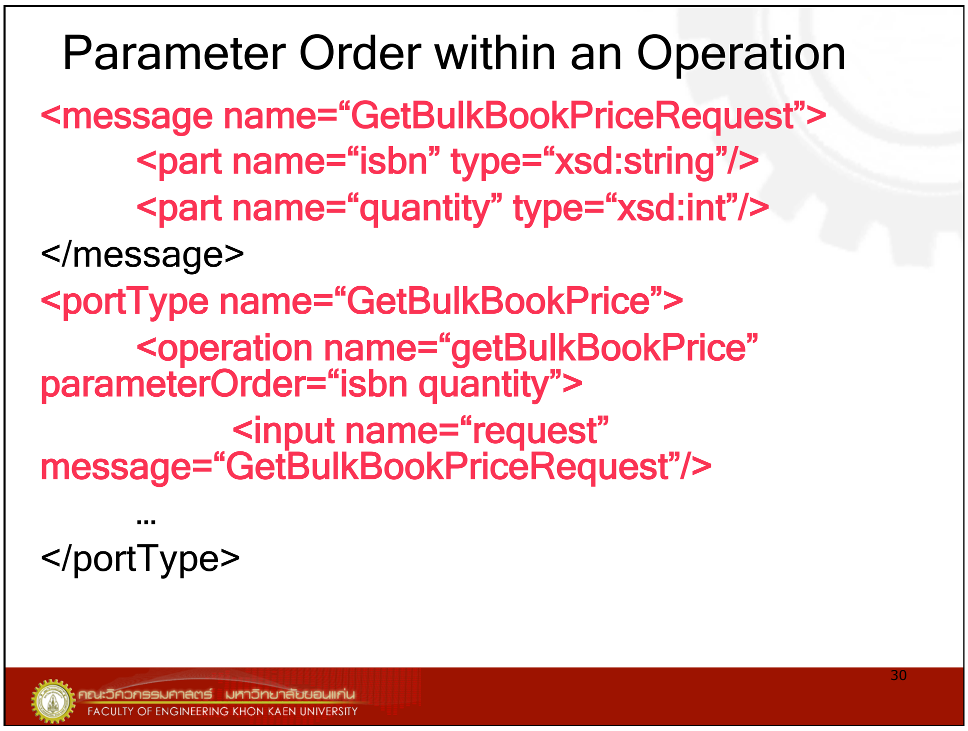
28



“operation” Element Example

```
<portType name=“BookQuote”>
        <operation name=“getBookPrice”>
                <input name=“isbn”
message=“mh:GetBookPriceRequest”/>
                <output name=“price”
message=“mh:GetBookPriceResponse”/>
                <fault
name=“InvalidArgumentFault”
message=“mh:InvalidArgumentFault”/>
        </operation>
</portType>
```

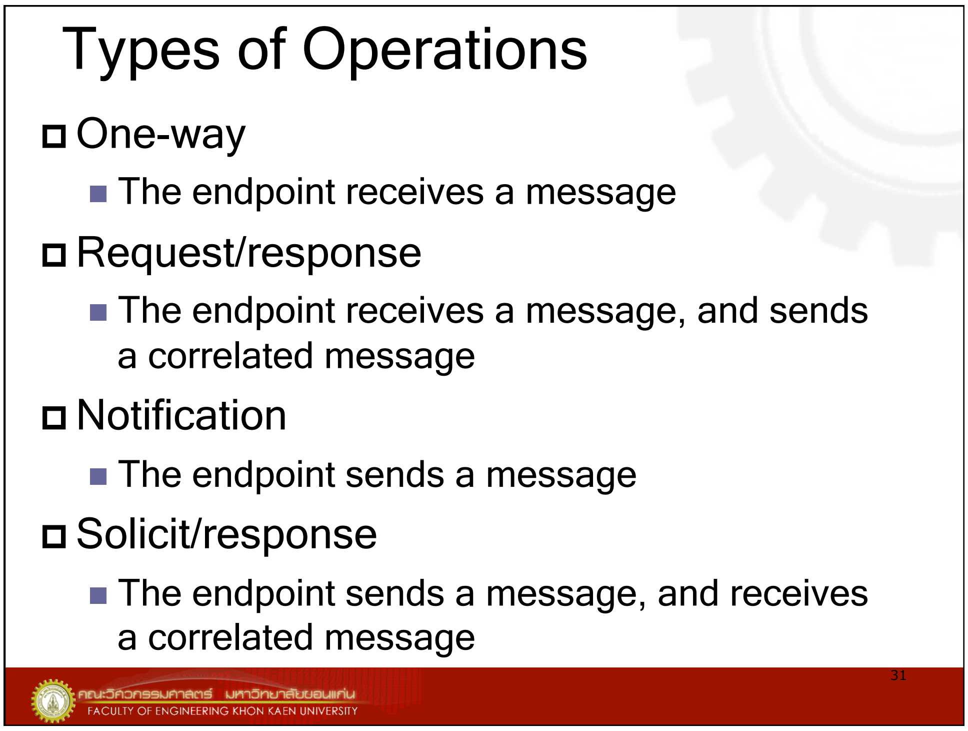
29



Parameter Order within an Operation

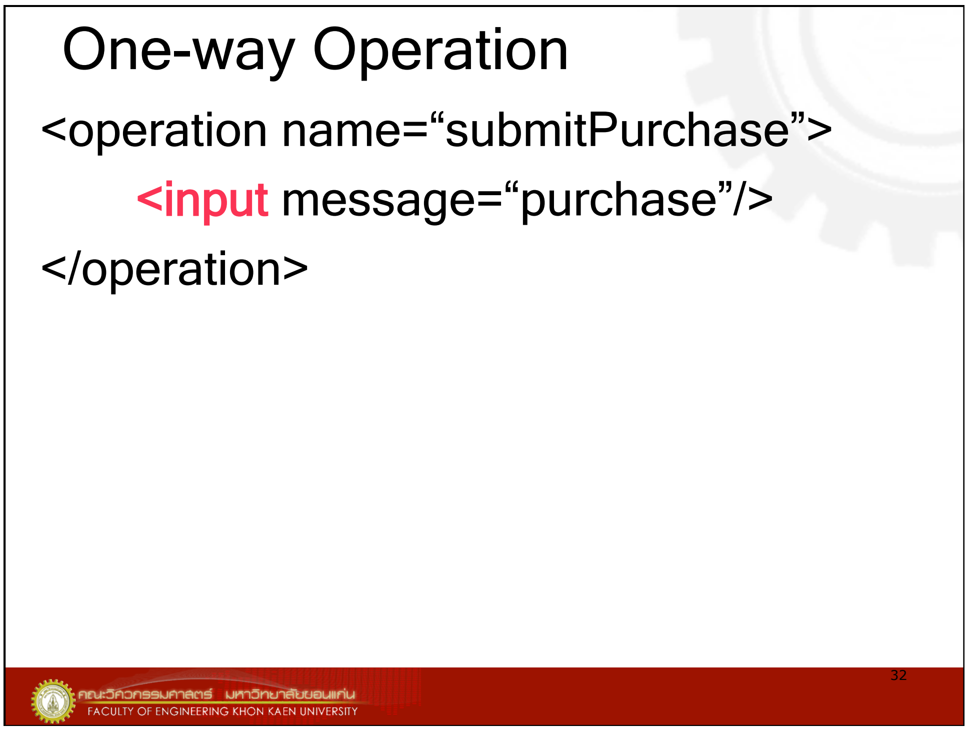
```
<message name=“GetBulkBookPriceRequest”>
        <part name=“isbn” type=“xsd:string”/>
        <part name=“quantity” type=“xsd:int”/>
</message>
<portType name=“GetBulkBookPrice”>
        <operation name=“getBulkBookPrice”
parameterOrder=“isbn quantity”>
                <input name=“request”
message=“GetBulkBookPriceRequest”/>
        …
</portType>
```

30



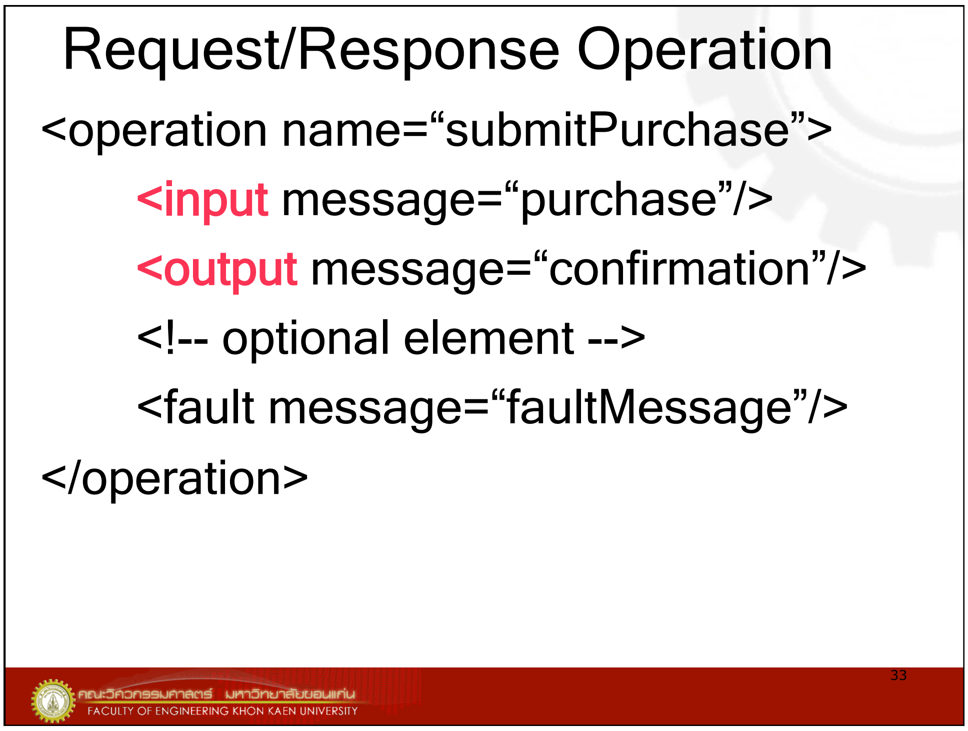
Types of Operations

- One-way
  - The endpoint receives a message
- Request/response
  - The endpoint receives a message, and sends a correlated message
- Notification
  - The endpoint sends a message
- Solicit/response
  - The endpoint sends a message, and receives a correlated message



One-way Operation

```
<operation name="submitPurchase">
      <input message="purchase"/>
</operation>
```

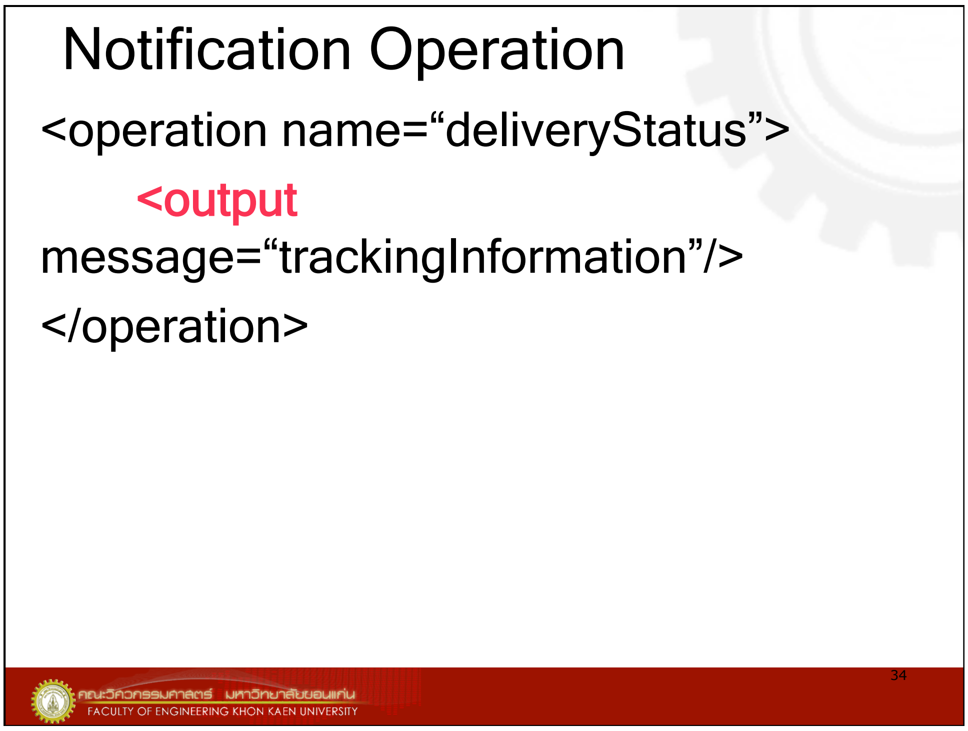


Request/Response Operation

```
<operation name=“submitPurchase”>
      <input message=“purchase”/>
      <output message=“confirmation”/>
      <!-- optional element -->
      <fault message=“faultMessage”/>
</operation>
```

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY

33

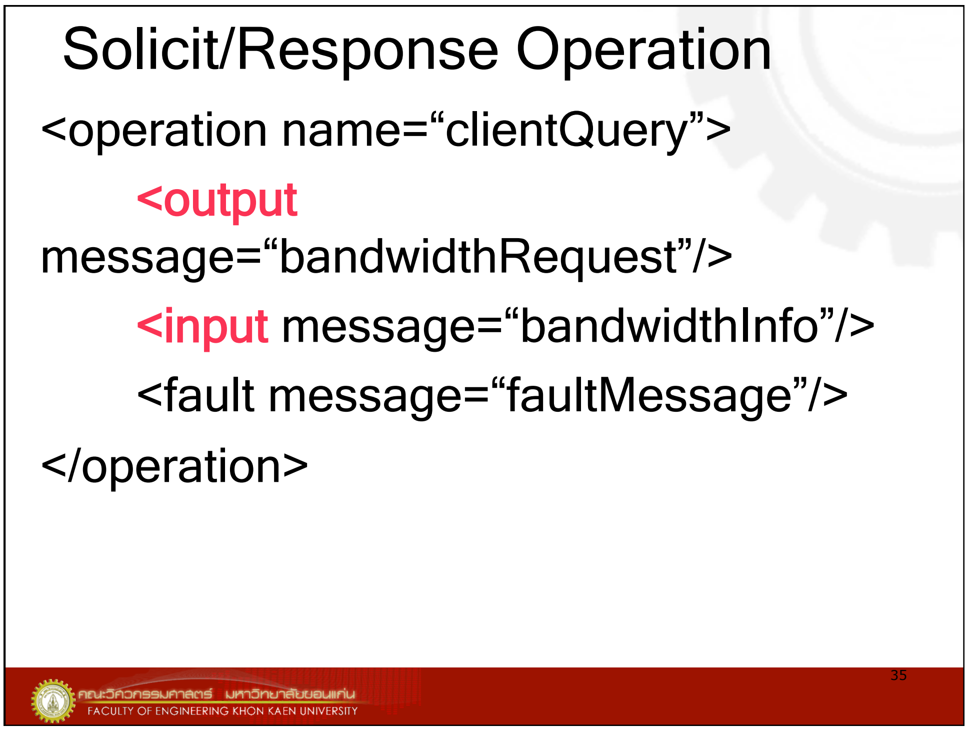


Notification Operation

```
<operation name=“deliveryStatus”>
      <output
message=“trackingInformation”/>
</operation>
```

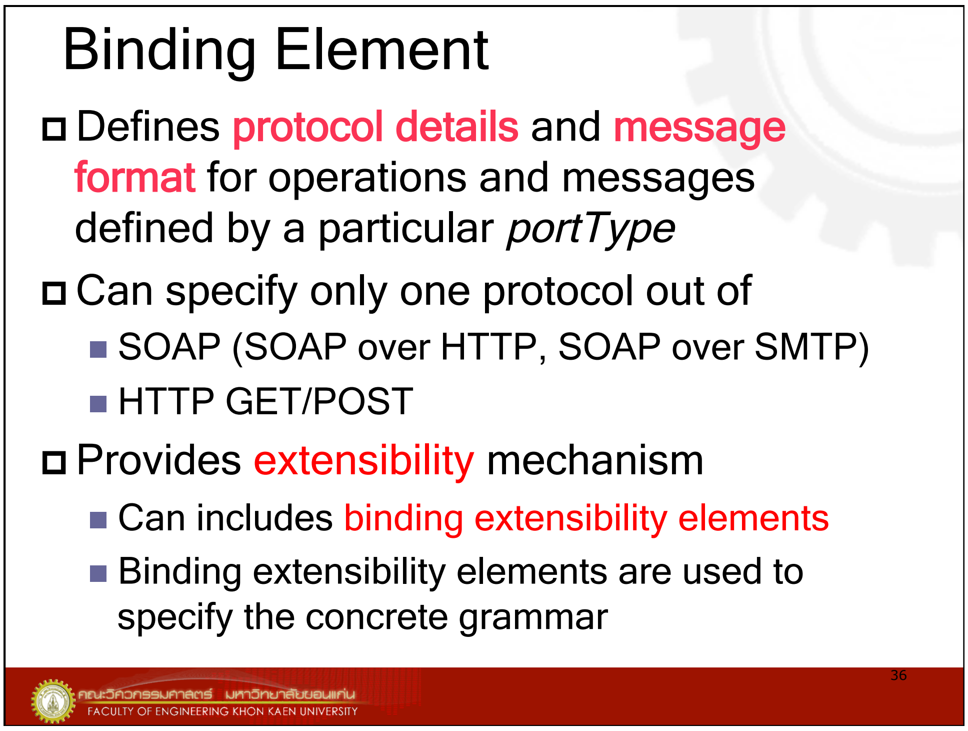
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY

34



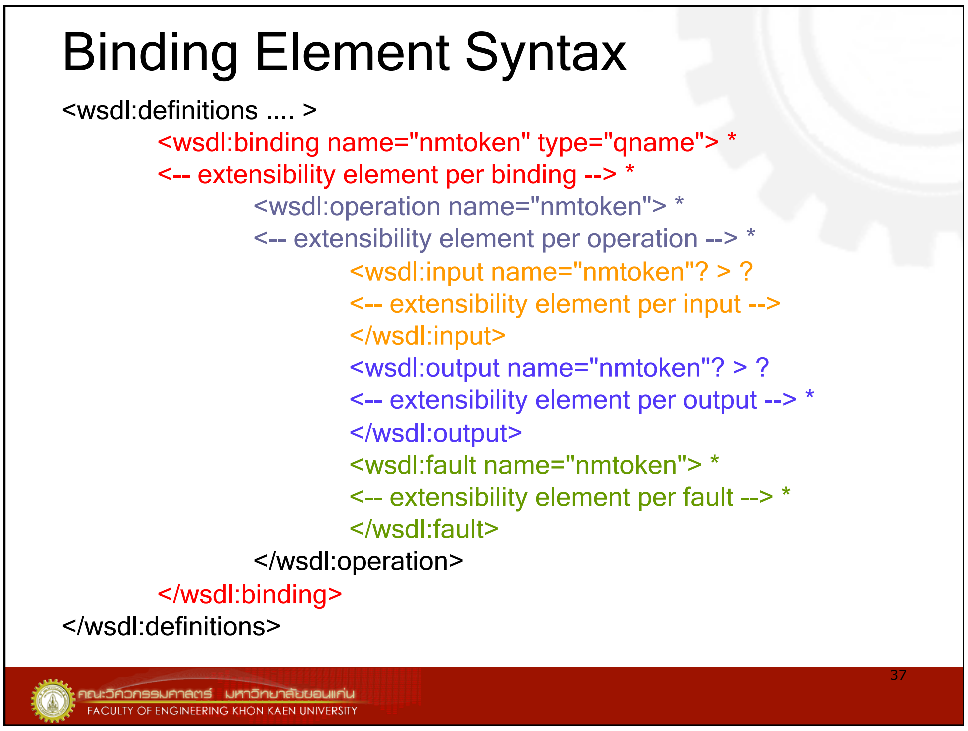
Solicit/Response Operation

```
<operation name="clientQuery">
    <output message="bandwidthRequest"/>
    <input message="bandwidthInfo"/>
    <fault message="faultMessage"/>
</operation>
```



Binding Element

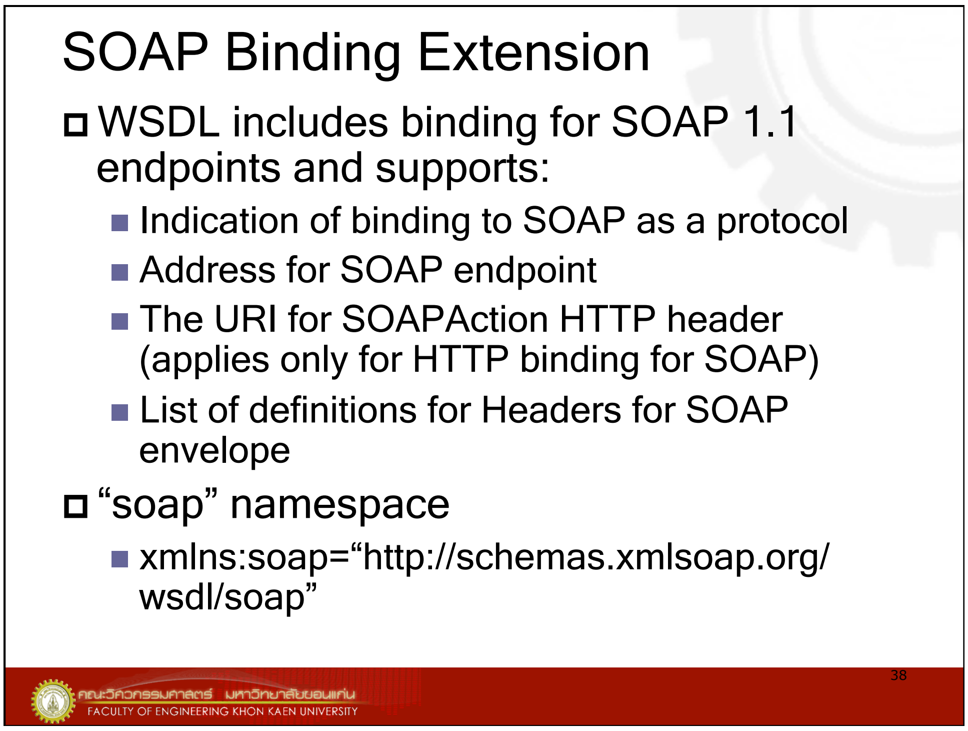
- Defines protocol details and message format for operations and messages defined by a particular *portType*
- Can specify only one protocol out of
  - SOAP (SOAP over HTTP, SOAP over SMTP)
  - HTTP GET/POST
- Provides extensibility mechanism
  - Can includes binding extensibility elements
  - Binding 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>
```

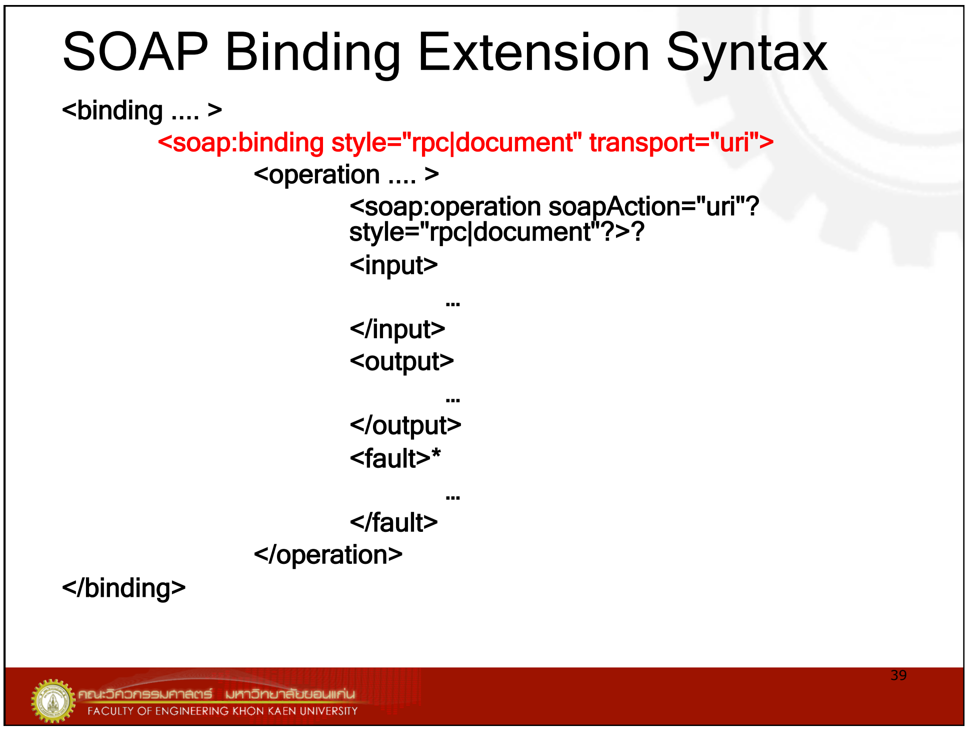
37



SOAP Binding Extension

- WSDL includes binding for SOAP 1.1 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 for SOAP)
  - List of definitions for Headers for SOAP envelope
- “soap” namespace
  - xmlns:soap=“http://schemas.xmlsoap.org/wsdl/soap”

38

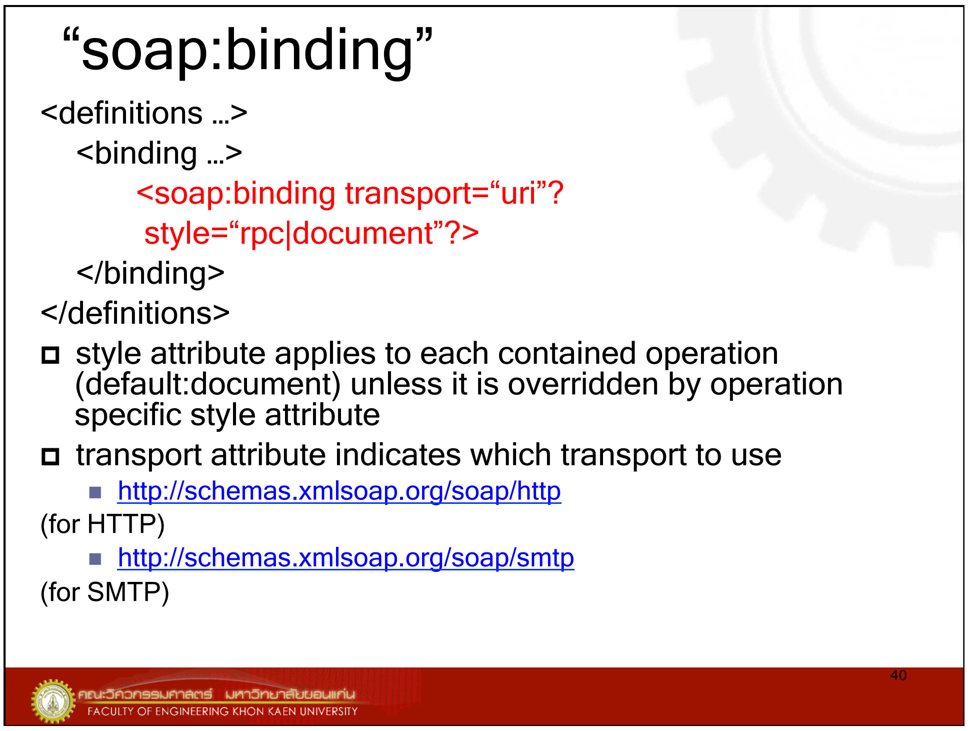


SOAP Binding Extension Syntax

```
<binding .... >
        <soap:binding style="rpc|document" transport="uri">
                <operation .... >
                        <soap:operation soapAction="uri"?
                        style="rpc|document"?>?
                        <input>
                                ...
                        </input>
                        <output>
                                ...
                        </output>
                        <fault>*
                                ...
                        </fault>
                </operation>
</binding>
```

39

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น FACULTY OF ENGINEERING KHON KAEN UNIVERSITY



“soap:binding”

```
<definitions …>
   <binding …>
        <soap:binding transport=“uri”?
         style=“rpc|document”?>
   </binding>
</definitions>
```

- style attribute applies to each contained operation (default:document) unless it is overridden by operation specific style attribute
- transport attribute indicates which transport to use
  - http://schemas.xmlsoap.org/soap/http

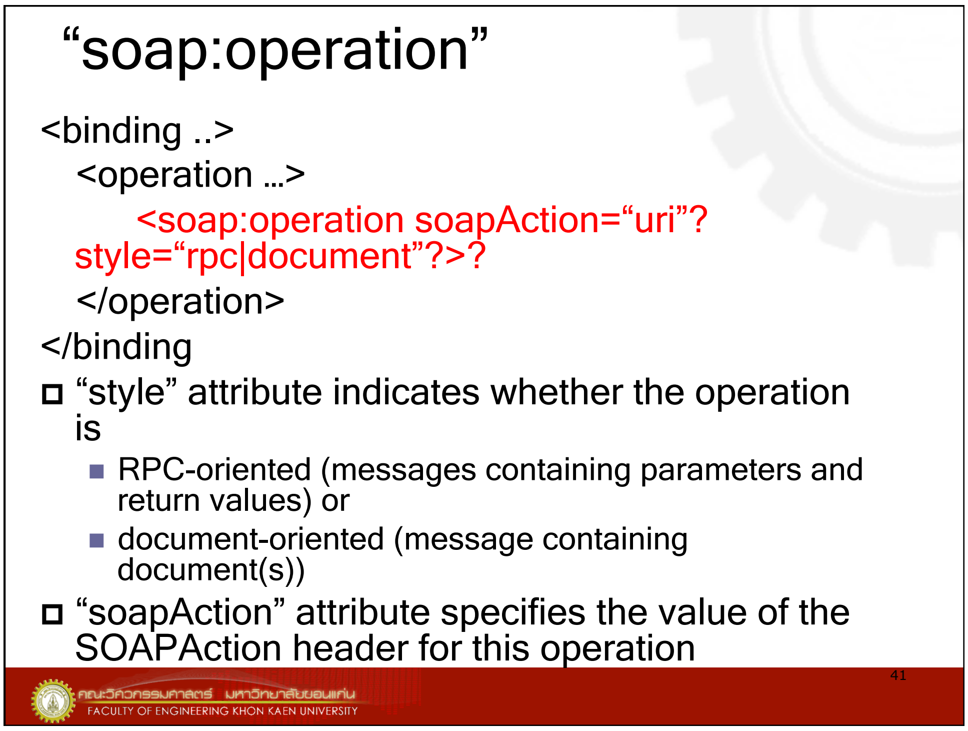
(for HTTP)

  - http://schemas.xmlsoap.org/soap/smtp

(for SMTP)

40

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น FACULTY OF ENGINEERING KHON KAEN UNIVERSITY



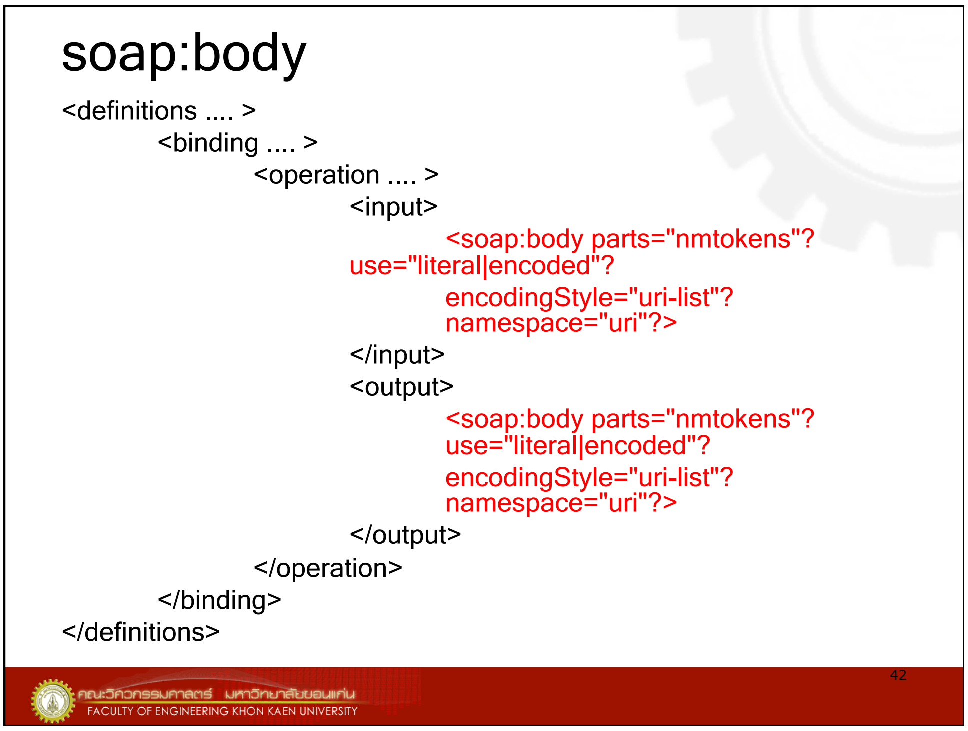
"soap:operation"

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

- "style" attribute indicates whether the operation is
  - RPC-oriented (messages containing parameters and return values) or
  - document-oriented (message containing document(s))
- "soapAction" attribute specifies the value of the SOAPAction header for this operation

41

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY

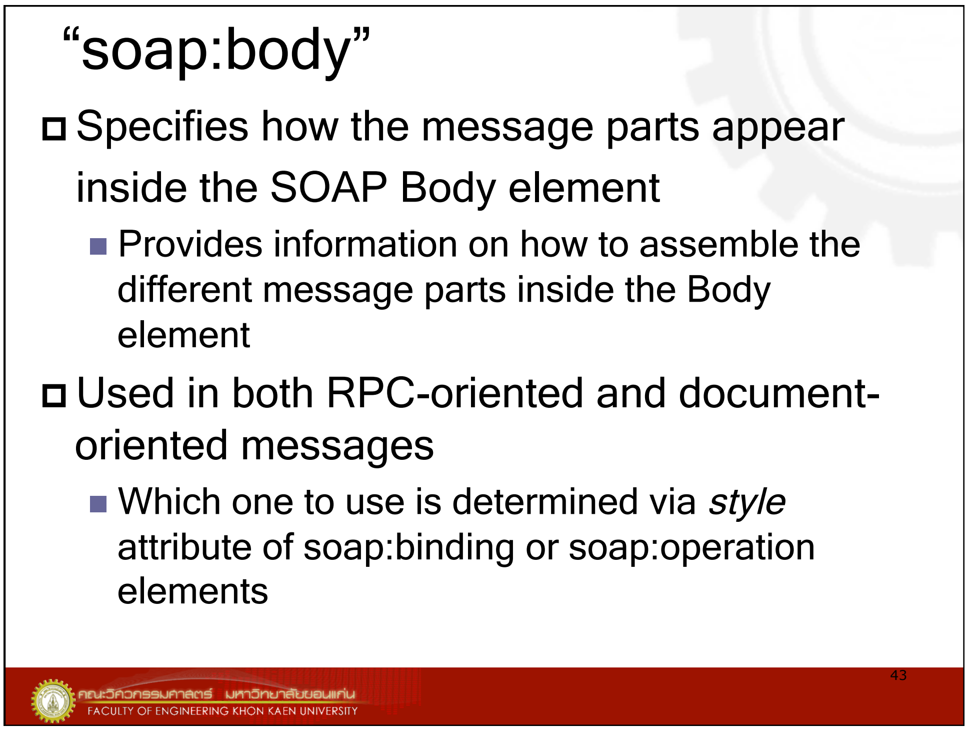


soap:body

```
<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>
```

42

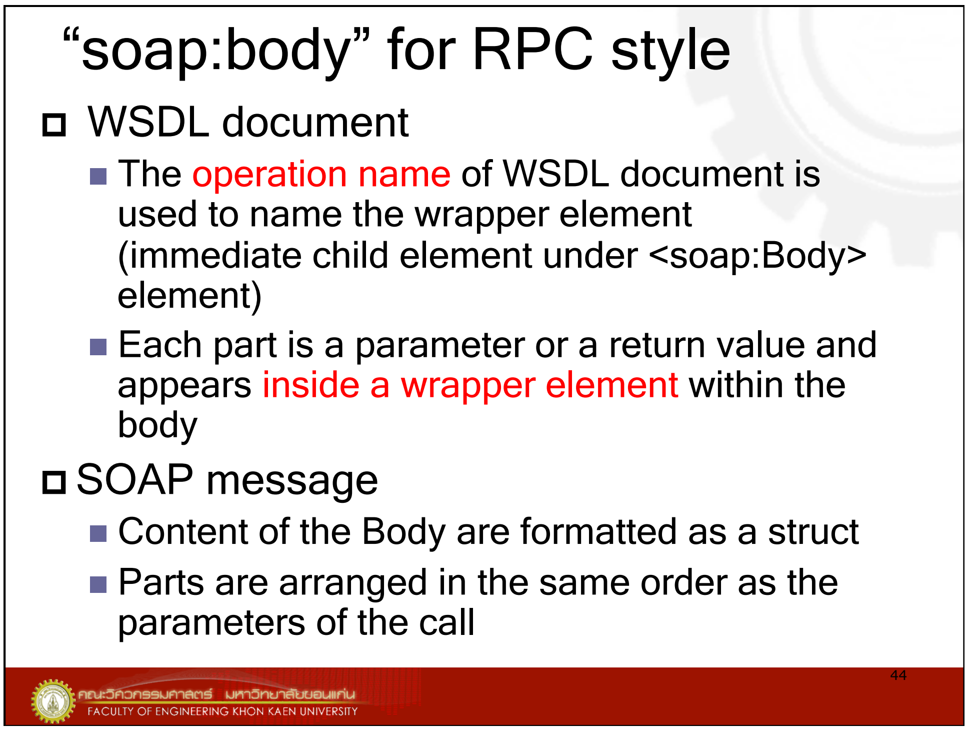
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY



“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

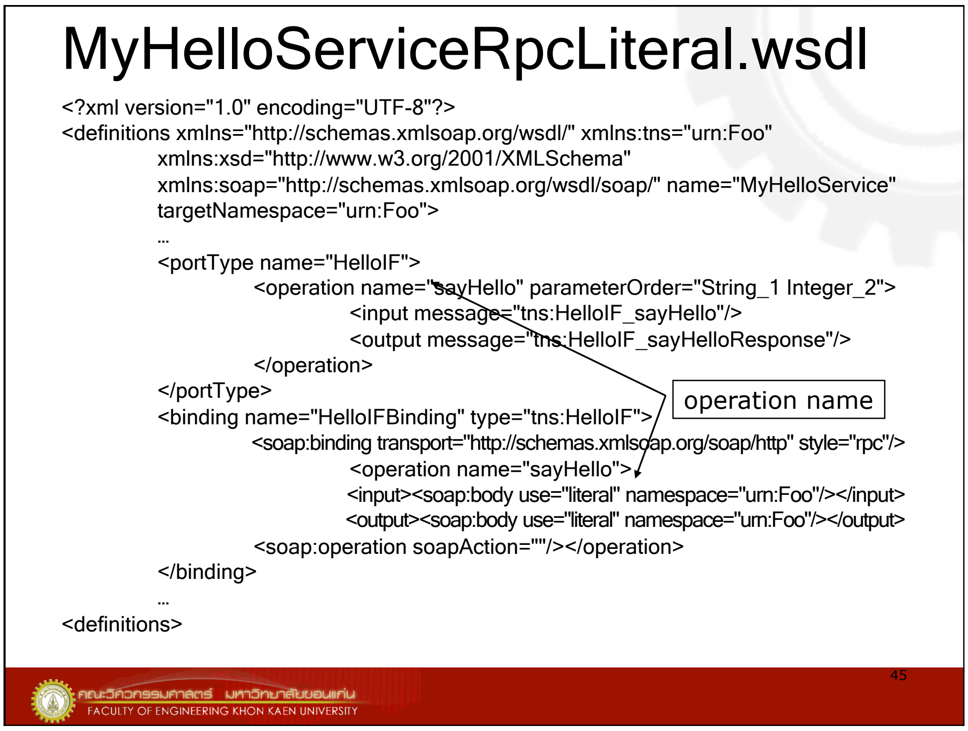
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY
43



“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)
  - Each part is a parameter or a return value and appears inside a wrapper element within the body
- SOAP message
  - Content of the Body are formatted as a struct
  - Parts are arranged in the same order as the parameters of the call

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY
44



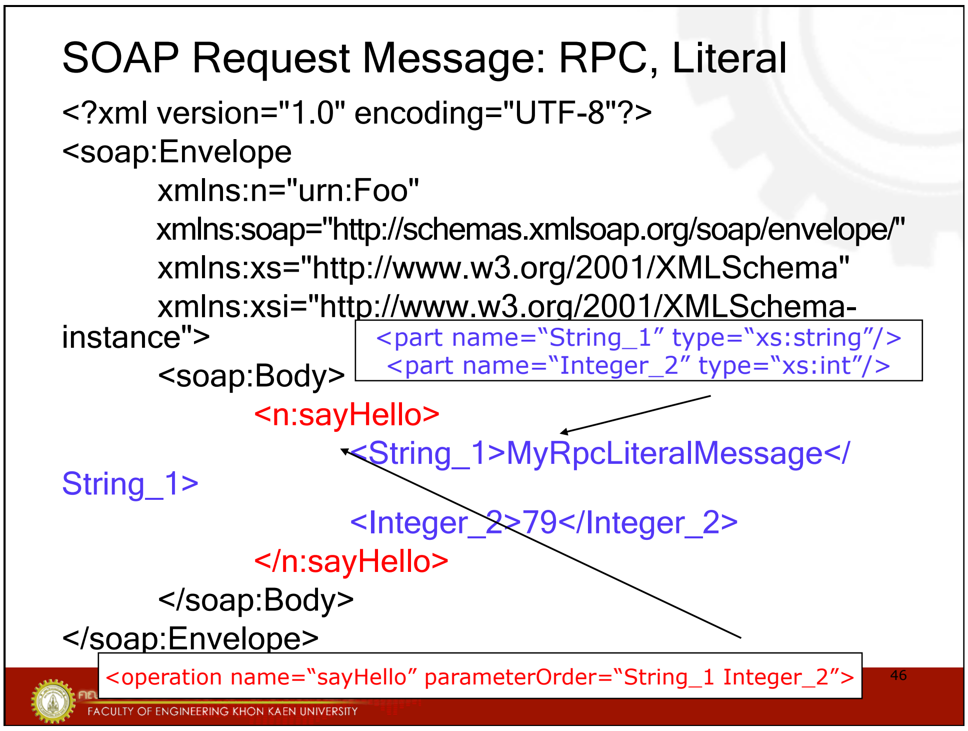
MyHelloServiceRpcLiteral.wsdl

```
<?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">
        …
        <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>
        …
<definitions>
```

operation name

45

FACULTY OF ENGINEERING KHON KAEN UNIVERSITY



SOAP Request Message: RPC, Literal

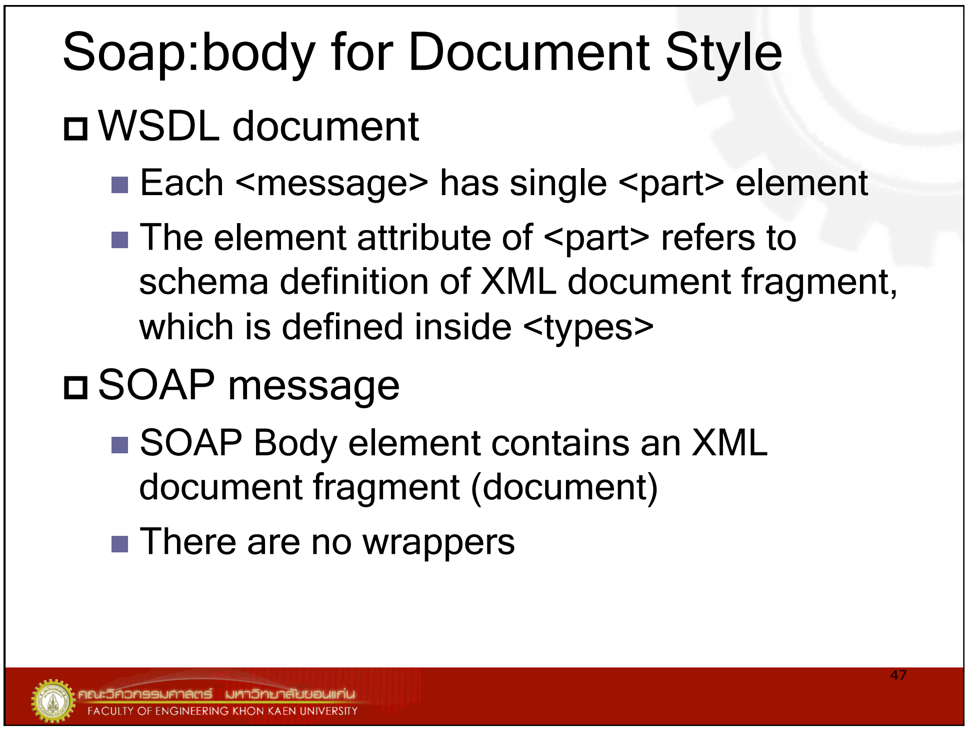
```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
        xmlns:n="urn:Foo"
        xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
        <soap:Body>
                <n:sayHello>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

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

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

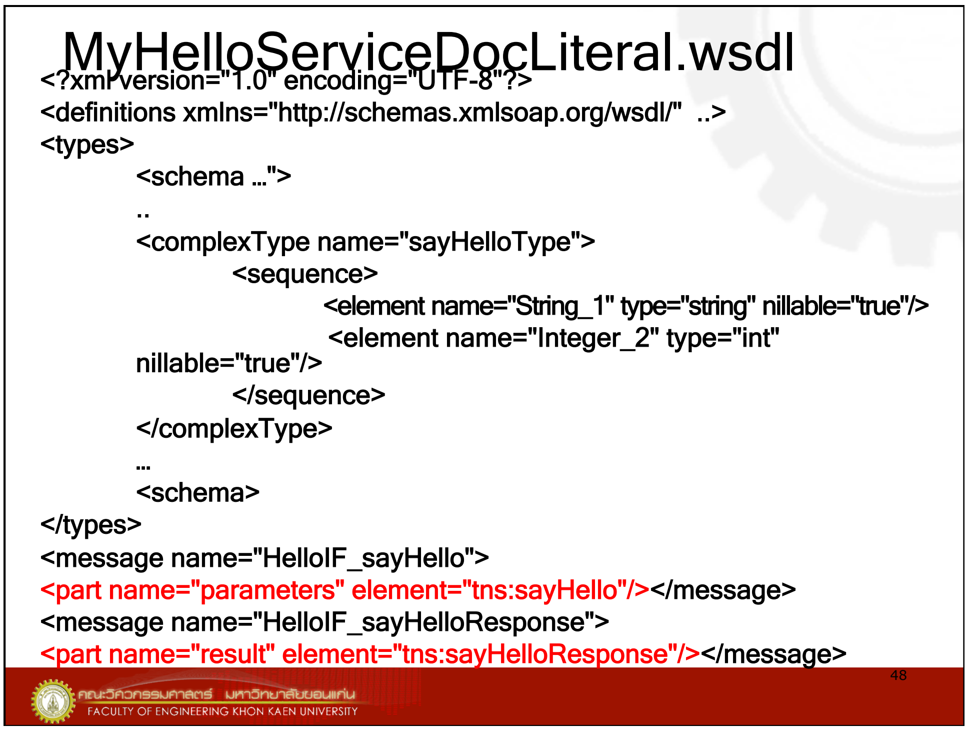
46

FACULTY OF ENGINEERING KHON KAEN UNIVERSITY



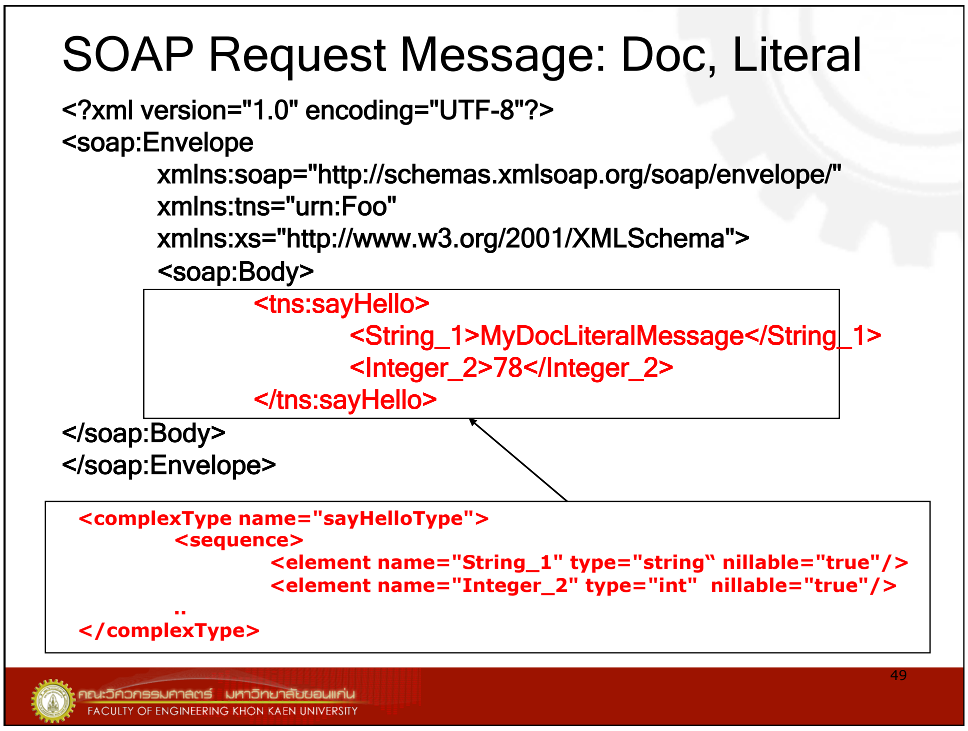
Soap:body for Document Style

- 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)
  - There are no wrappers



MyHelloServiceDocLiteral.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"  ..>
<types>
        <schema …">
        ..
        <complexType name="sayHelloType">
                <sequence>
                        <element name="String_1" type="string" nillable="true"/>
                         <element name="Integer_2" type="int"
        nillable="true"/>
                </sequence>
        </complexType>
        …
        <schema>
</types>
<message name="HelloIF_sayHello">
<part name="parameters" element="tns:sayHello"/></message>
<message name="HelloIF_sayHelloResponse">
<part name="result" element="tns:sayHelloResponse"/></message>
```



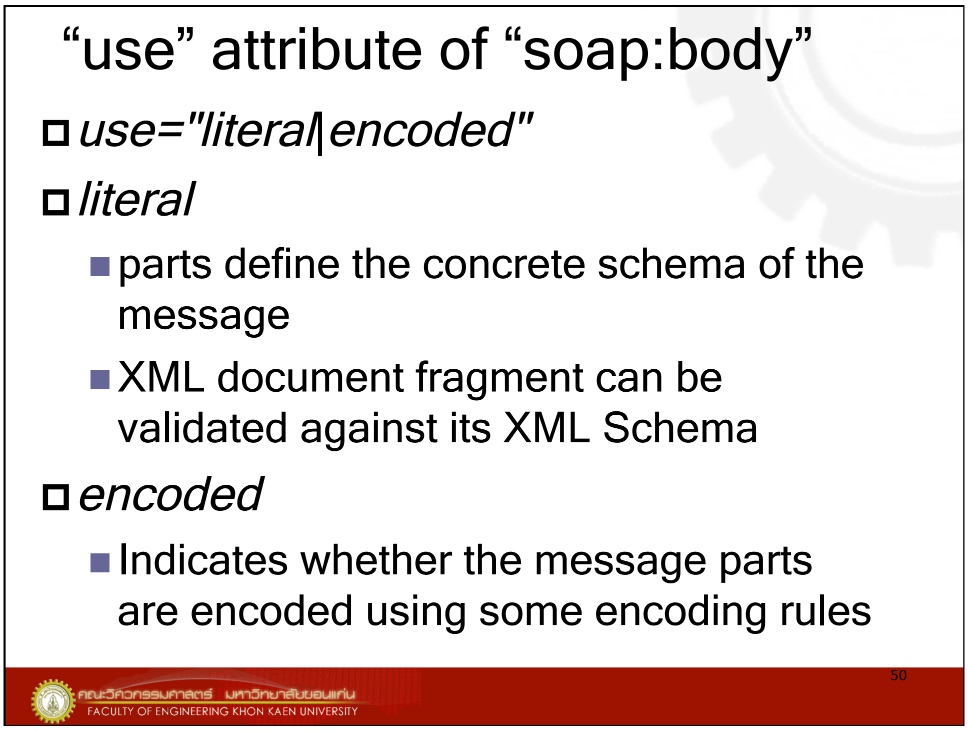
SOAP Request Message: Doc, Literal

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

49

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY

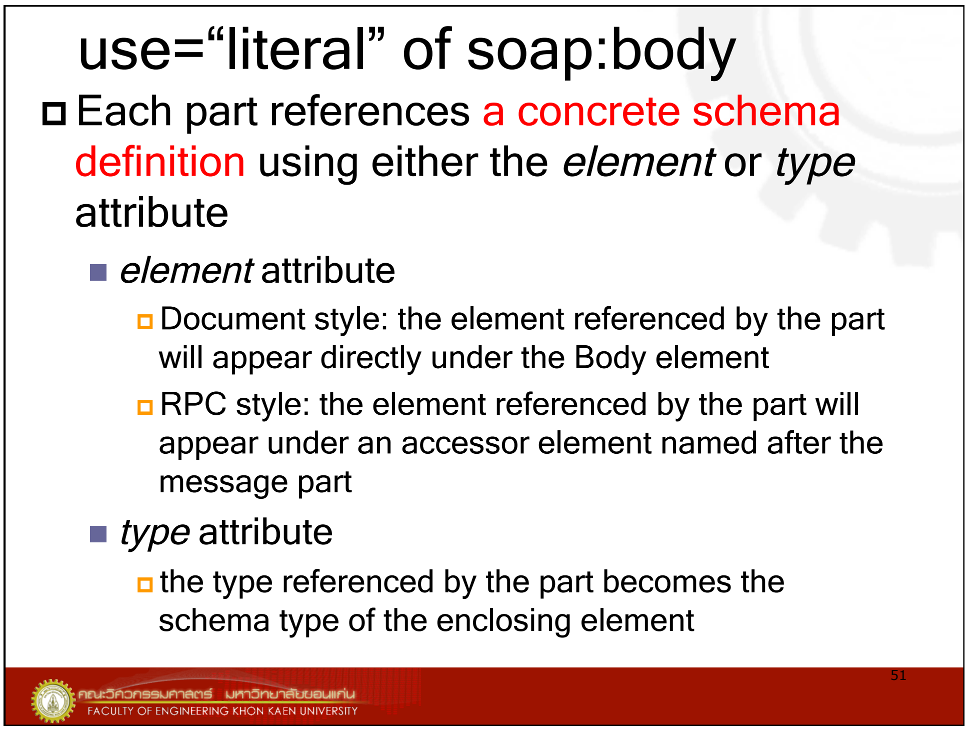


“use” attribute of “soap:body”

- *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

50

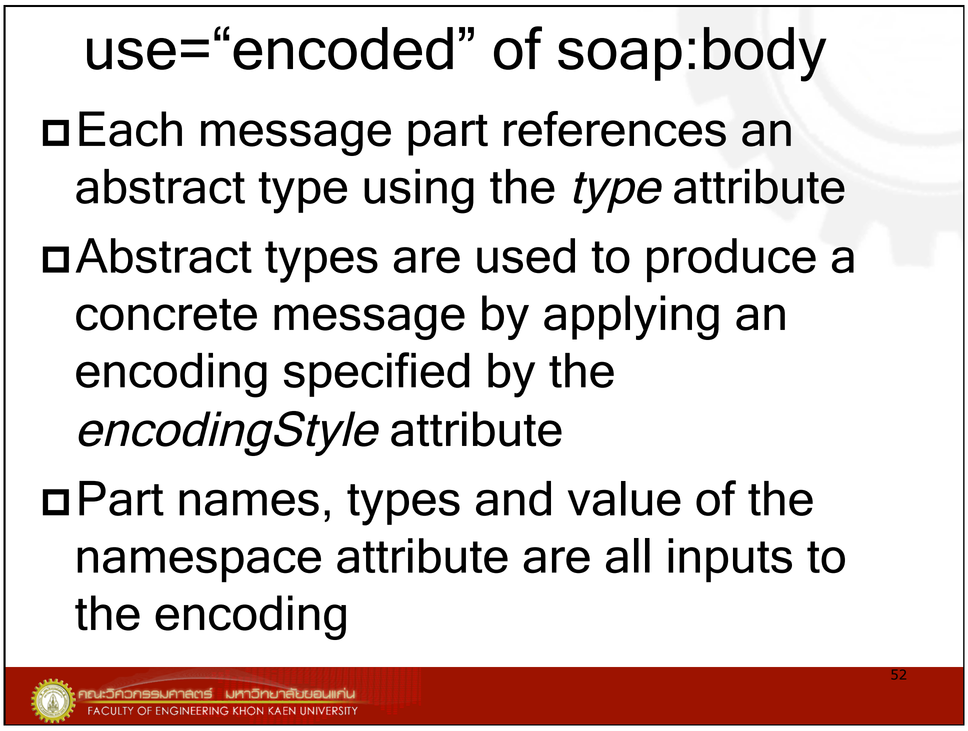
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY



use="literal" of soap:body

- Each part references a concrete schema definition using either the *element* or *type* attribute
  - *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

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 51



use="encoded" of soap:body

- 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

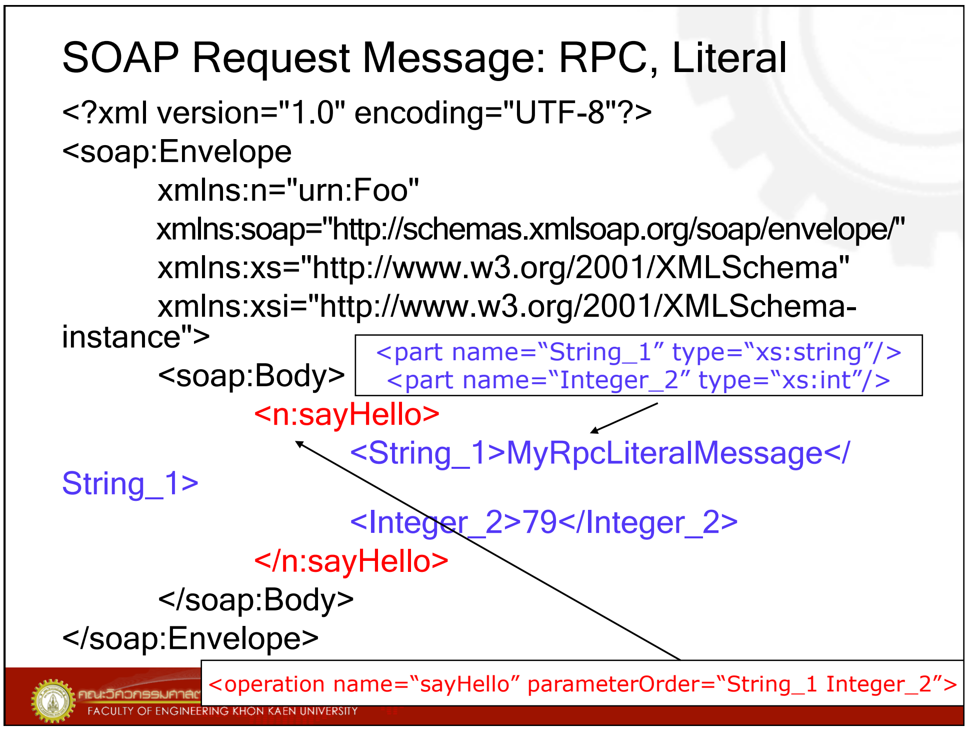
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 52

Possible Style/Use Combinations

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

MyHelloServiceRpcLiteral.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions  ...>
 ...
<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>
...
<definitions>
```

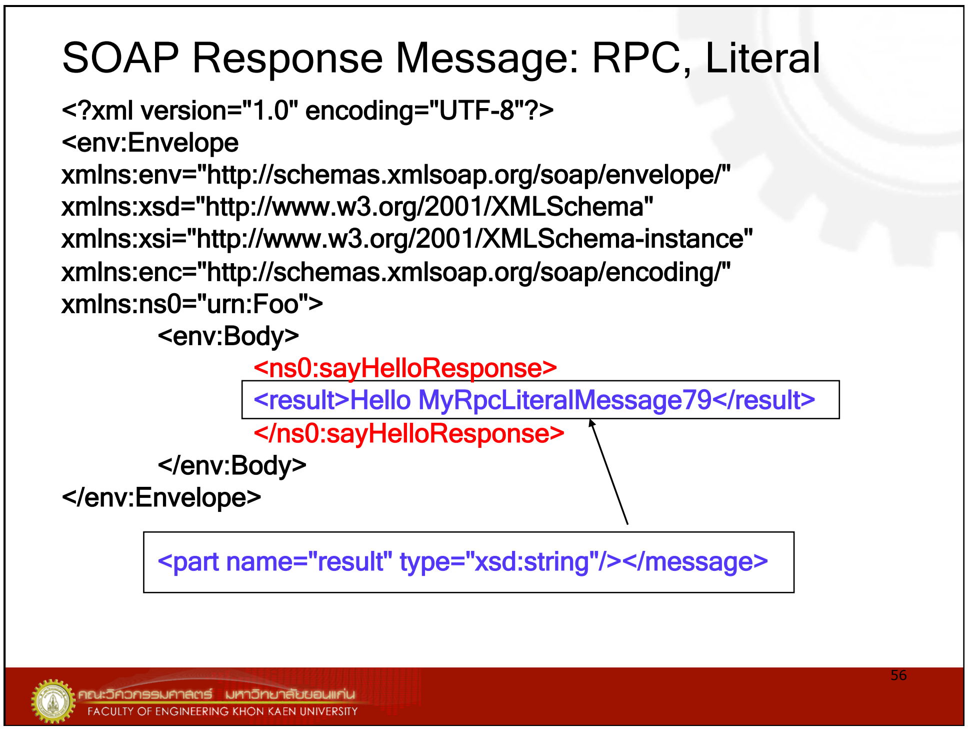


SOAP Request Message: RPC, Literal

```
<?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="xs:string"/>
<part name="Integer_2" type="xs:int"/>
```

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

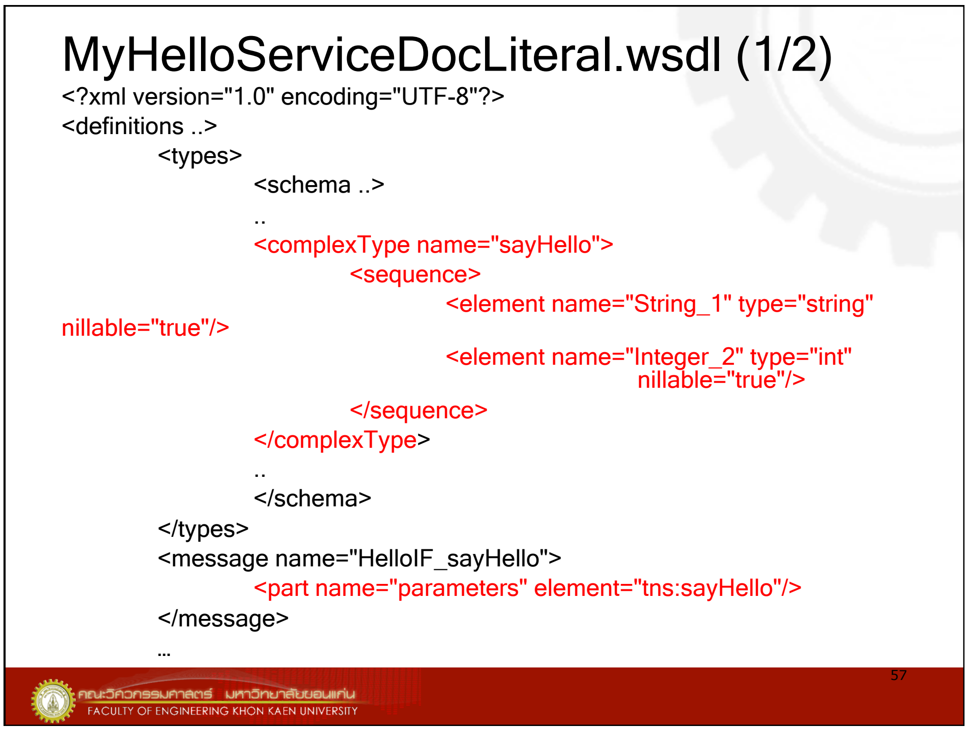


SOAP Response Message: RPC, Literal

```
<?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>
```

```
<part name="result" type="xsd:string"/></message>
```

56

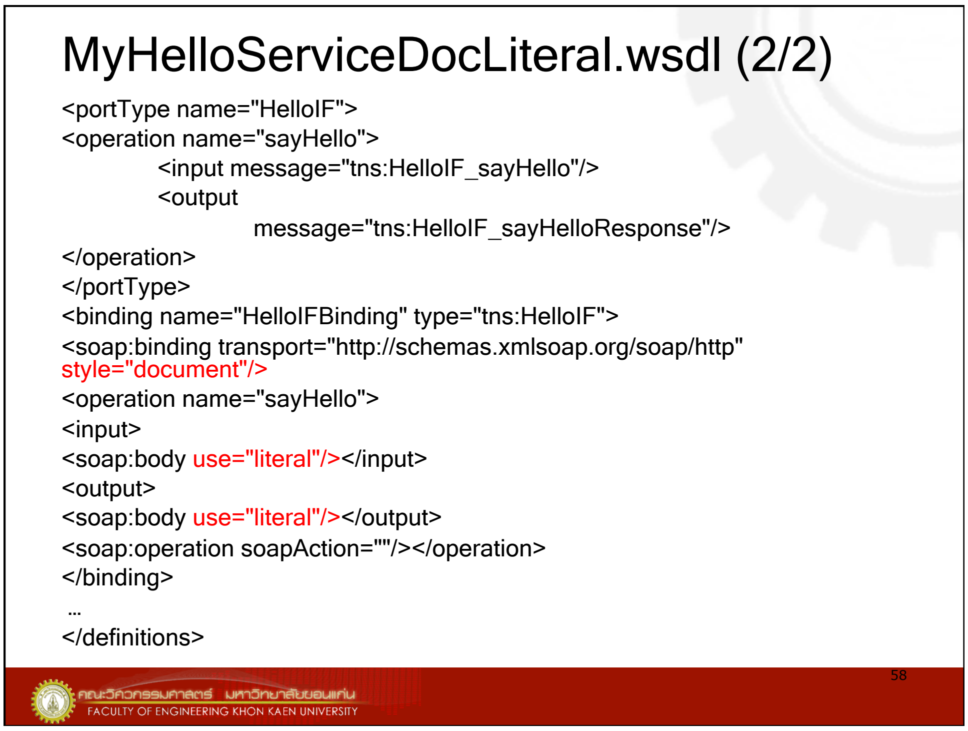


MyHelloServiceDocLiteral.wsdl (1/2)

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions ..>
        <types>
                <schema ..>
                ..
                <complexType name="sayHello">
                        <sequence>
                                <element name="String_1" type="string"
nillable="true"/>
                                <element name="Integer_2" type="int"
                                                nillable="true"/>
                        </sequence>
                </complexType>
                ..
                </schema>
        </types>
        <message name="HelloIF_sayHello">
                <part name="parameters" element="tns:sayHello"/>
        </message>
        …
```

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY

57

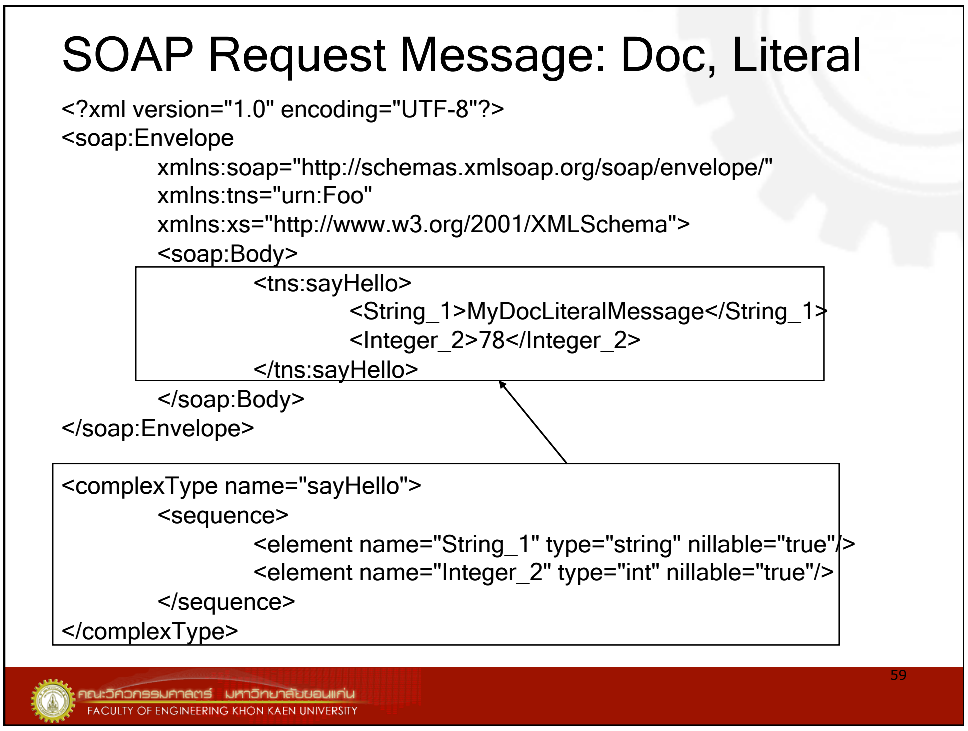


MyHelloServiceDocLiteral.wsdl (2/2)

```
<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>
 …
</definitions>
```

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY

58



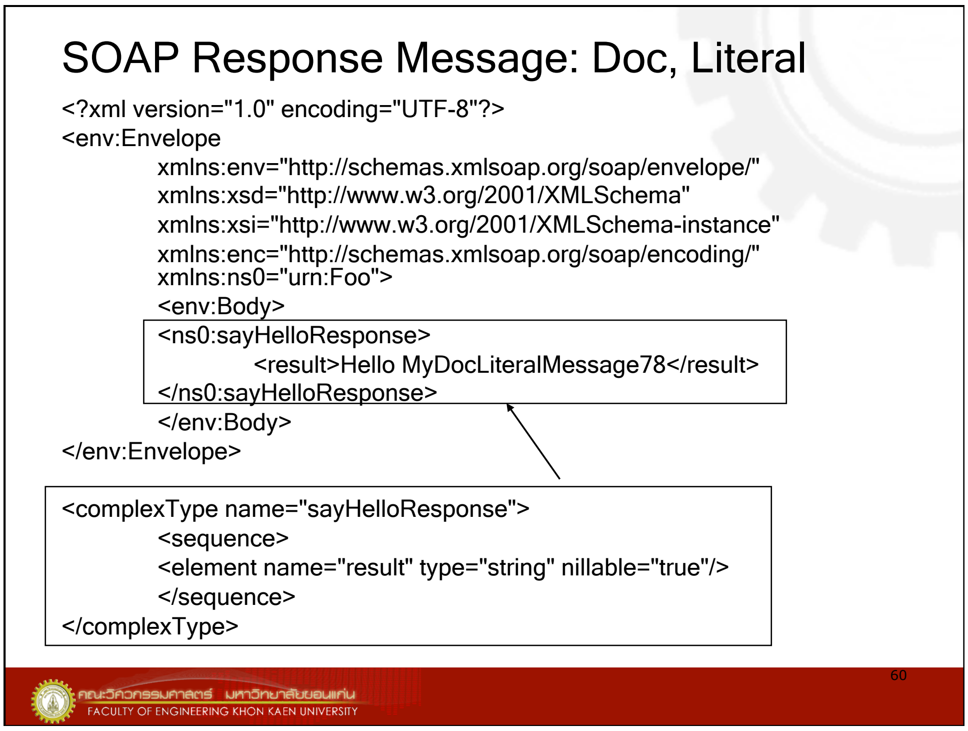
SOAP Request Message: Doc, Literal

```
<?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>
```

59

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY



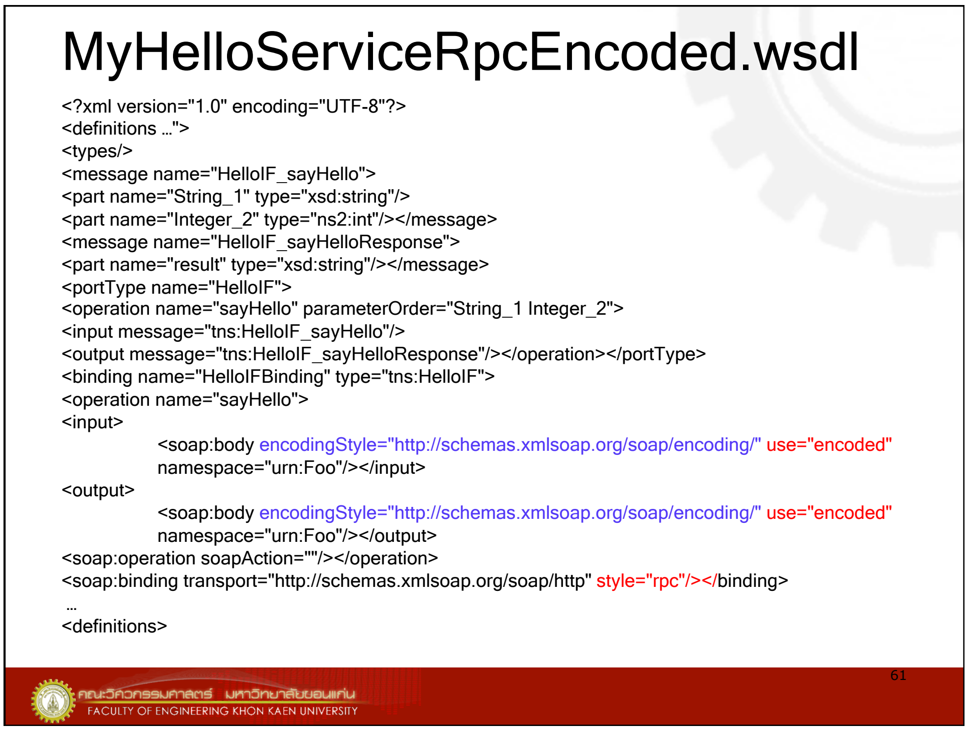
SOAP Response Message: Doc, Literal

```
<?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>
```

60

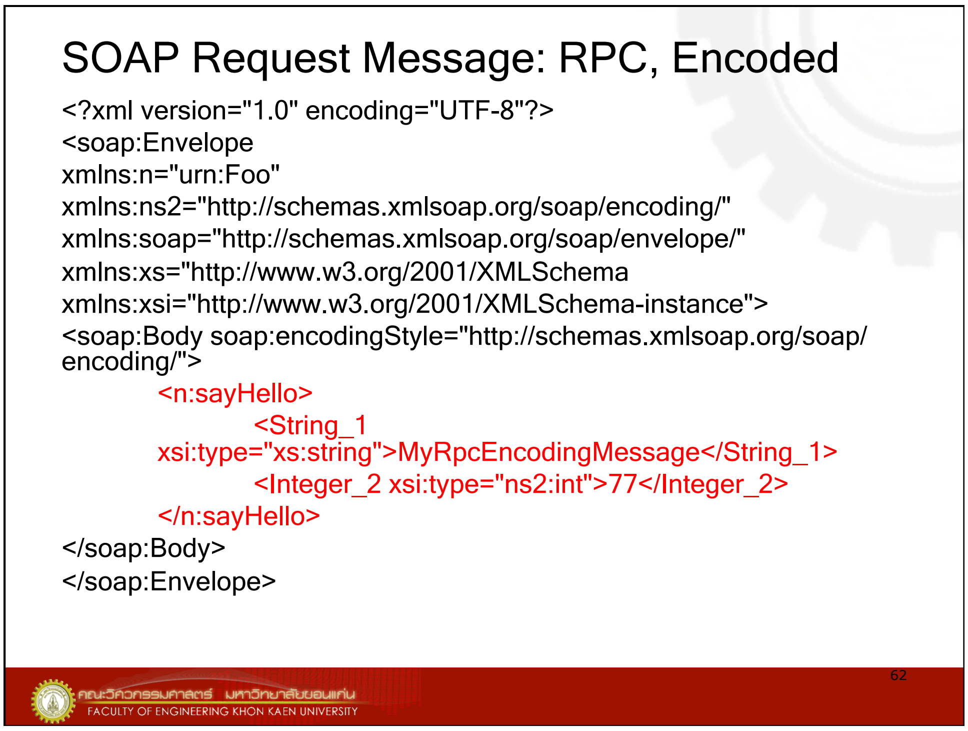
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY



MyHelloServiceRpcEncoded.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions …">
<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>
…
<definitions>
```

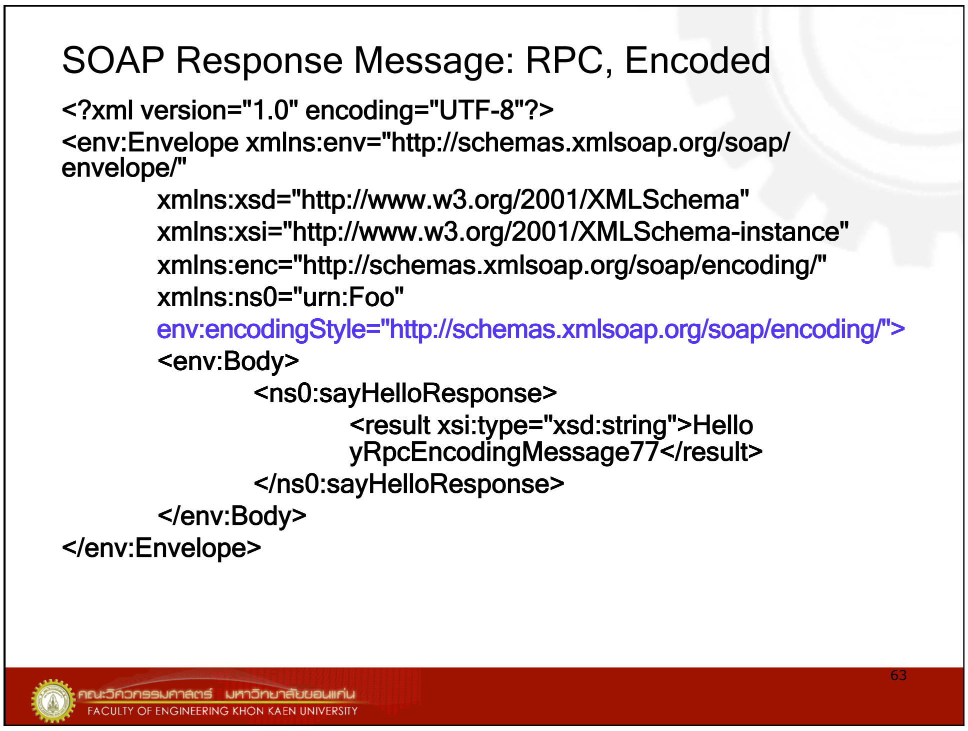
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY
61



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>
```

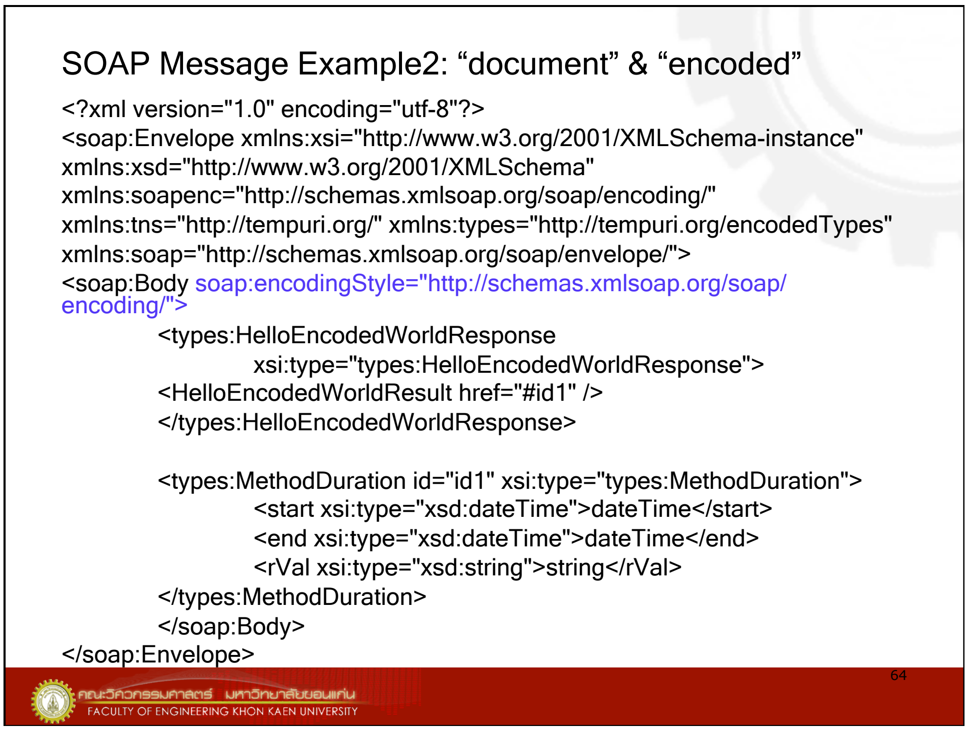
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY
62



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
                        yRpcEncodingMessage77</result>
                </ns0:sayHelloResponse>
        </env:Body>
</env:Envelope>
```

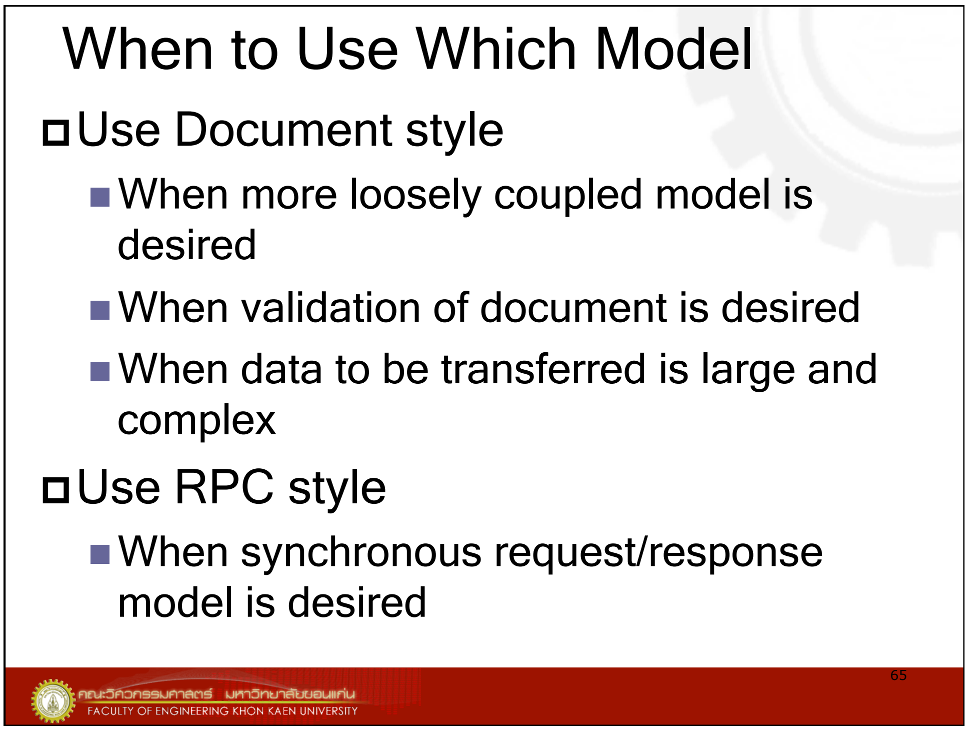
63



SOAP Message Example2: “document” & “encoded”

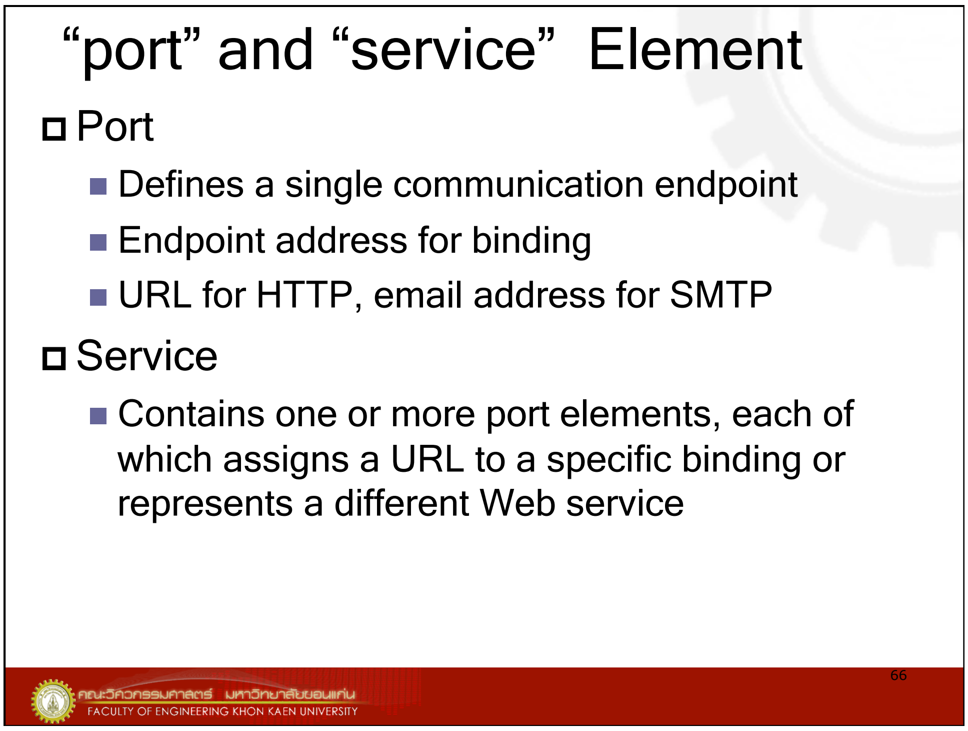
```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="http://tempuri.org/" xmlns:types="http://tempuri.org/encodedTypes"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">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

64



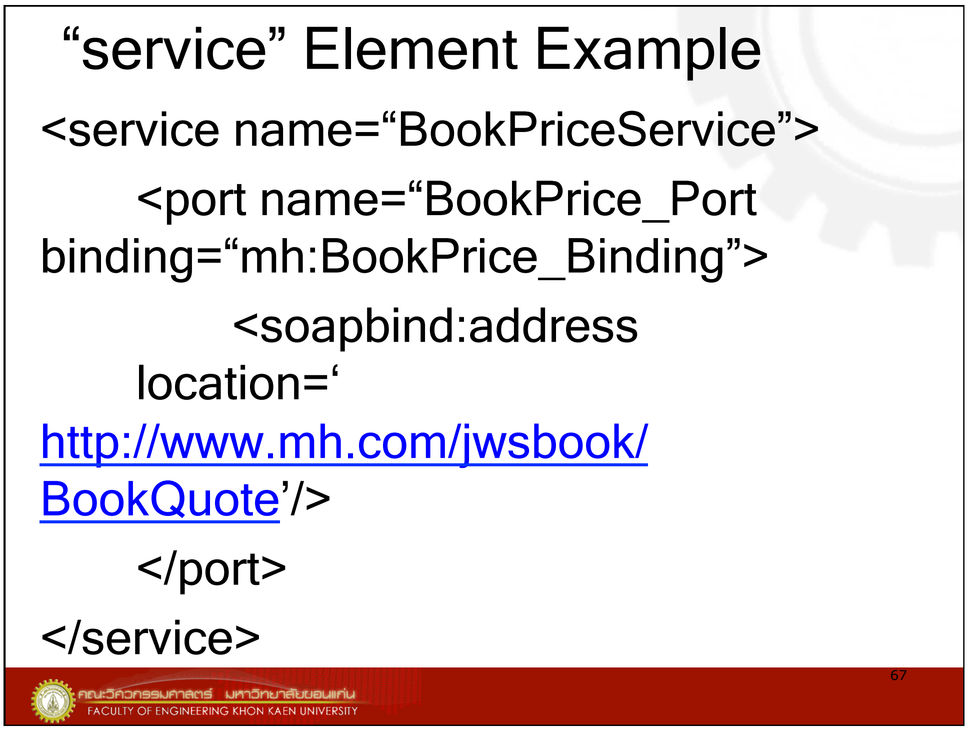
When to Use Which Model

- Use Document style
  - When more loosely coupled model is desired
  - When validation of document is desired
  - When data to be transferred is large and complex
- Use RPC style
  - When synchronous request/response model is desired



“port” and “service” Element

- Port
  - Defines a single communication endpoint
  - Endpoint address for binding
  - URL for HTTP, email address for SMTP
- Service
  - Contains one or more port elements, each of which assigns a URL to a specific binding or represents a different Web service

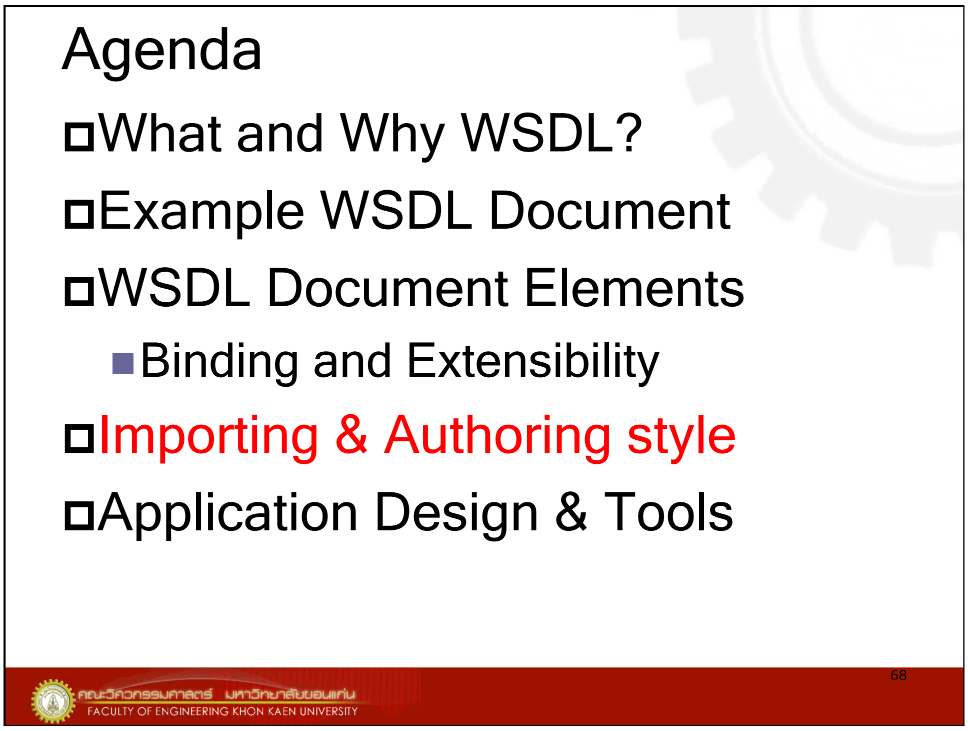


“service” Element Example

```
<service name=“BookPriceService”>
        <port name=“BookPrice_Port
binding=“mh:BookPrice_Binding”>
                <soapbind:address
        location=‘
http://www.mh.com/jwsbook/
BookQuote’/>
        </port>
</service>
```

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY

67

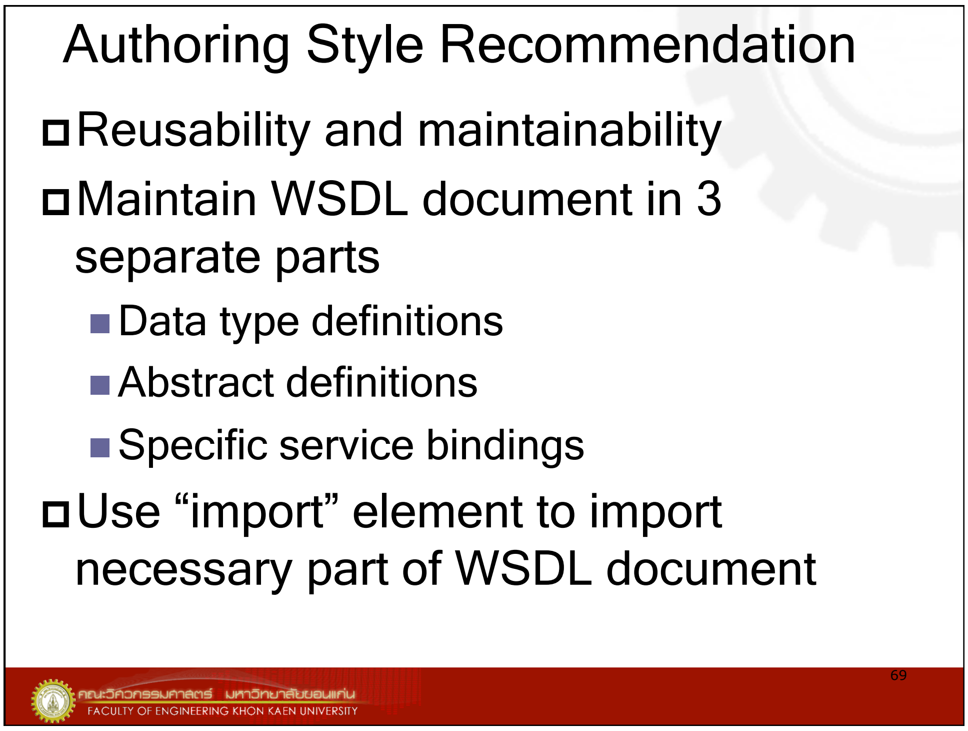


Agenda

- What and Why WSDL?
- Example WSDL Document
- WSDL Document Elements
  - Binding and Extensibility
- Importing & Authoring style
- Application Design & Tools

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY

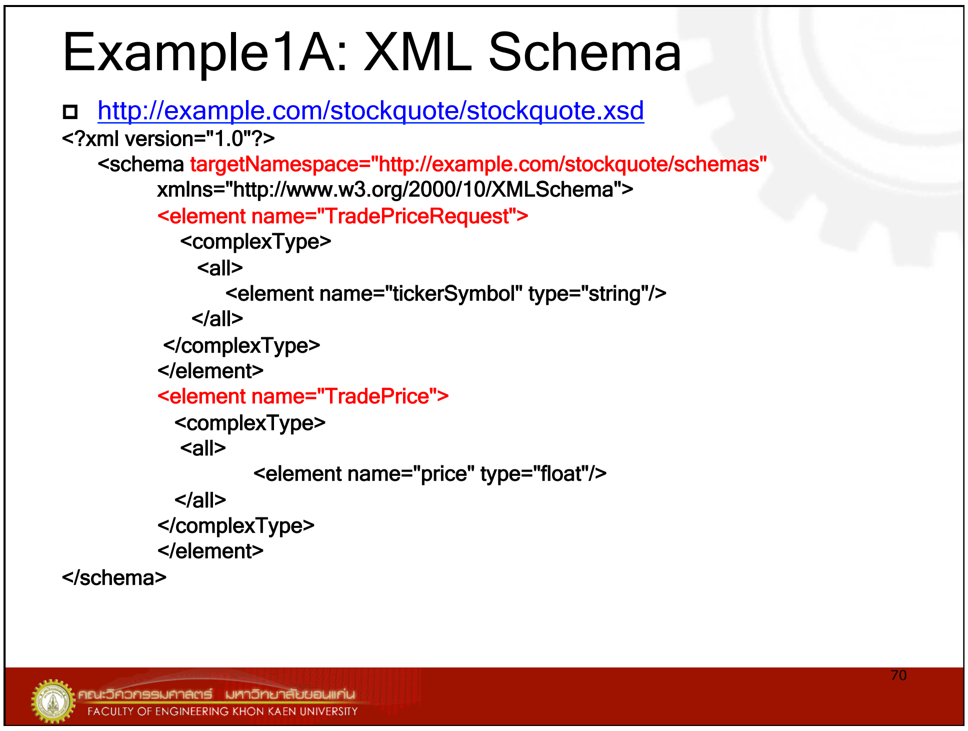
68



Authoring Style Recommendation

- Reusability and maintainability
- Maintain WSDL document in 3 separate parts
  - Data type definitions
  - Abstract definitions
  - Specific service bindings
- Use “import” element to import necessary part of WSDL document

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY
69

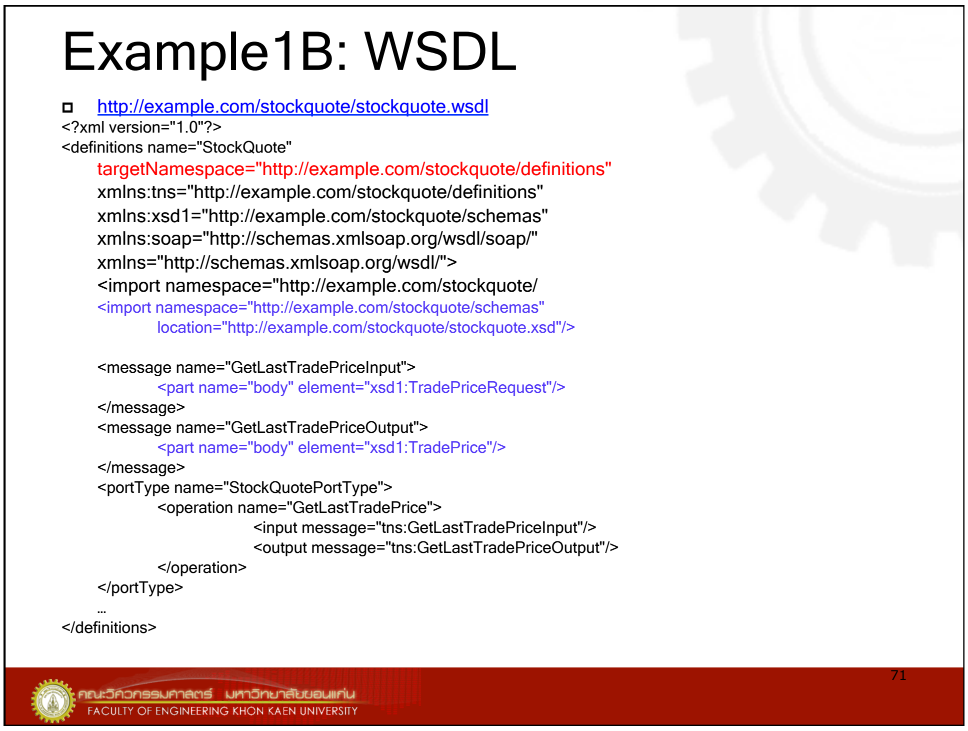


Example1A: XML Schema

- http://example.com/stockquote/stockquote.xsd

```
<?xml version="1.0"?>
    <schema targetNamespace="http://example.com/stockquote/schemas"
            xmlns="http://www.w3.org/2000/10/XMLSchema">
            <element name="TradePriceRequest">
               <complexType>
                 <all>
                     <element name="tickerSymbol" type="string"/>
                 </all>
             </complexType>
            </element>
            <element name="TradePrice">
              <complexType>
               <all>
                        <element name="price" type="float"/>
              </all>
            </complexType>
            </element>
</schema>
```

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY
70



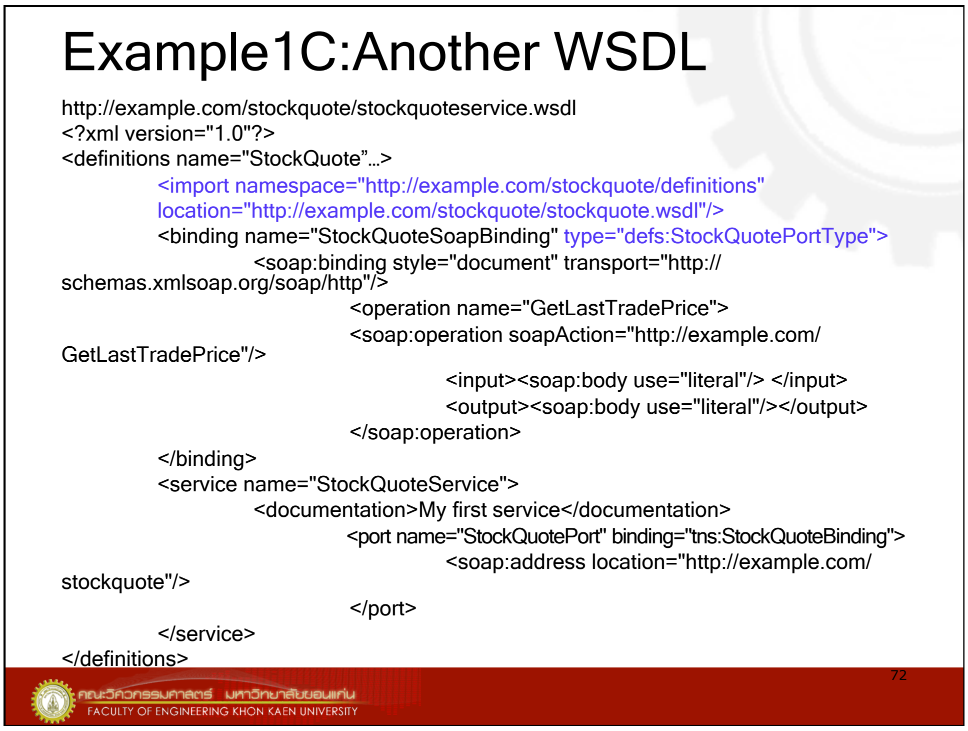
Example1B: WSDL

- http://example.com/stockquote/stockquote.wsdl

```
<?xml version="1.0"?>
<definitions name="StockQuote"
      targetNamespace="http://example.com/stockquote/definitions"
      xmlns:tns="http://example.com/stockquote/definitions"
      xmlns:xsd1="http://example.com/stockquote/schemas"
      xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
      xmlns="http://schemas.xmlsoap.org/wsdl/">
      <import namespace="http://example.com/stockquote/
      <import namespace="http://example.com/stockquote/schemas"
            location="http://example.com/stockquote/stockquote.xsd"/>

      <message name="GetLastTradePriceInput">
            <part name="body" element="xsd1:TradePriceRequest"/>
      </message>
      <message name="GetLastTradePriceOutput">
            <part name="body" element="xsd1:TradePrice"/>
      </message>
      <portType name="StockQuotePortType">
            <operation name="GetLastTradePrice">
                  <input message="tns:GetLastTradePriceInput"/>
                  <output message="tns:GetLastTradePriceOutput"/>
            </operation>
      </portType>
      …
</definitions>
```

FACULTY OF ENGINEERING KHON KAEN UNIVERSITY

71

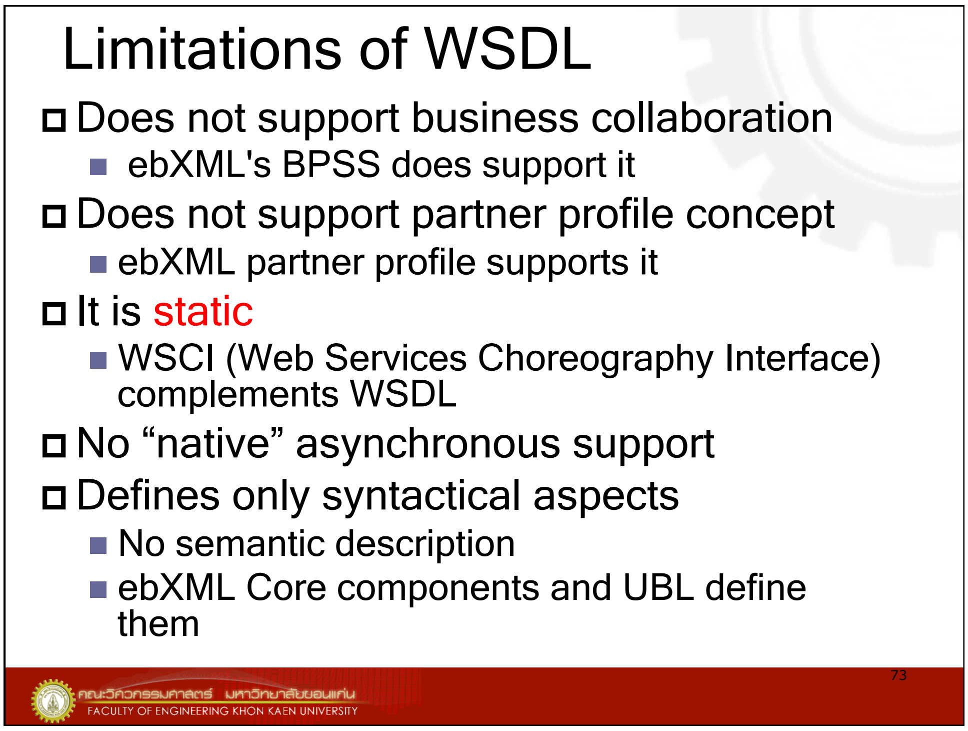


Example1C:Another WSDL

```
http://example.com/stockquote/stockquoteservice.wsdl
<?xml version="1.0"?>
<definitions name="StockQuote"…>
      <import namespace="http://example.com/stockquote/definitions"
      location="http://example.com/stockquote/stockquote.wsdl"/>
      <binding name="StockQuoteSoapBinding" type="defs:StockQuotePortType">
            <soap:binding style="document" transport="http://
schemas.xmlsoap.org/soap/http"/>
                  <operation name="GetLastTradePrice">
                  <soap:operation soapAction="http://example.com/
GetLastTradePrice"/>
                        <input><soap:body use="literal"/> </input>
                        <output><soap:body use="literal"/></output>
                  </soap:operation>
      </binding>
      <service name="StockQuoteService">
            <documentation>My first service</documentation>
                  <port name="StockQuotePort" binding="tns:StockQuoteBinding">
                        <soap:address location="http://example.com/
stockquote"/>
                  </port>
      </service>
</definitions>
```

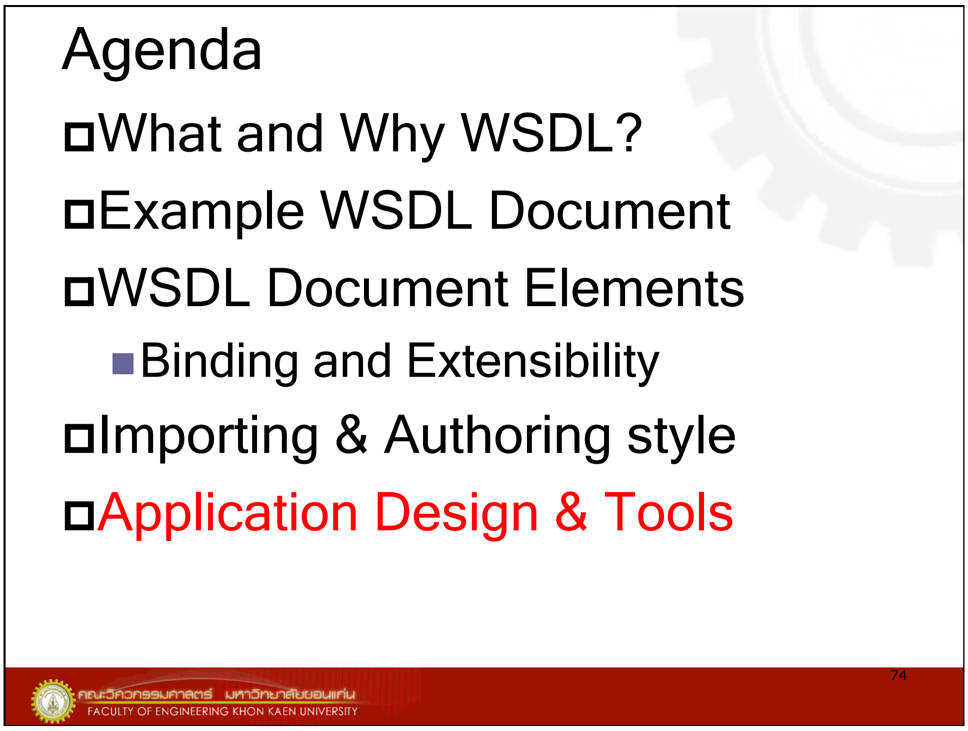
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY

72



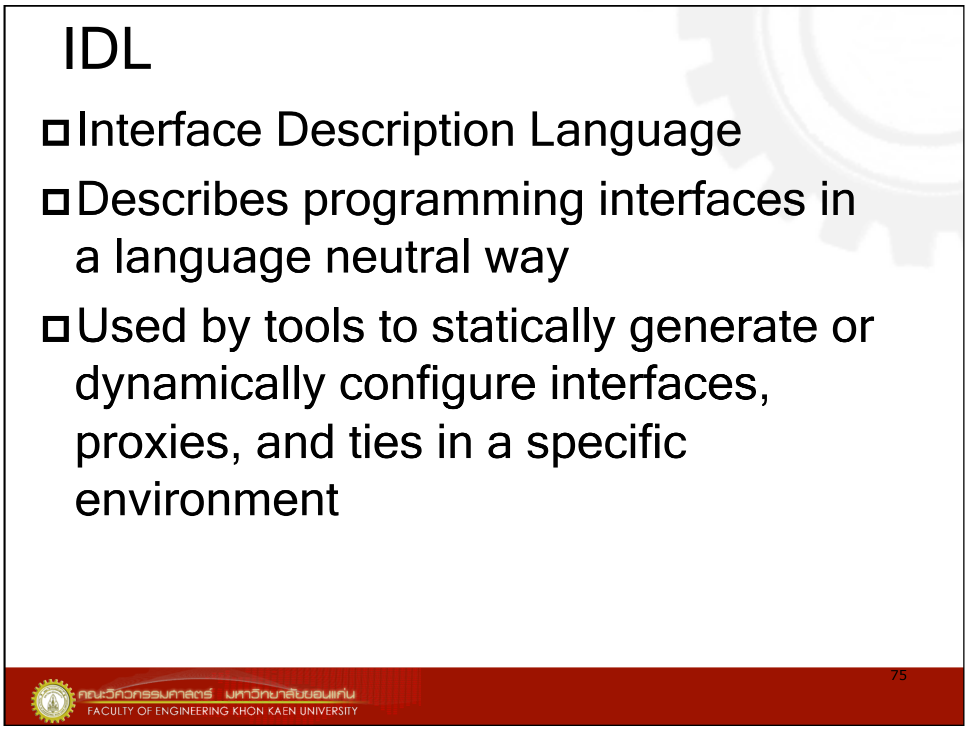
Limitations of WSDL

- Does not support business collaboration
  - ebXML's BPSS does support it
- Does not support partner profile concept
  - ebXML partner profile supports it
- It is static
  - WSCI (Web Services Choreography Interface) complements WSDL
- No “native” asynchronous support
- Defines only syntactical aspects
  - No semantic description
  - ebXML Core components and UBL define them



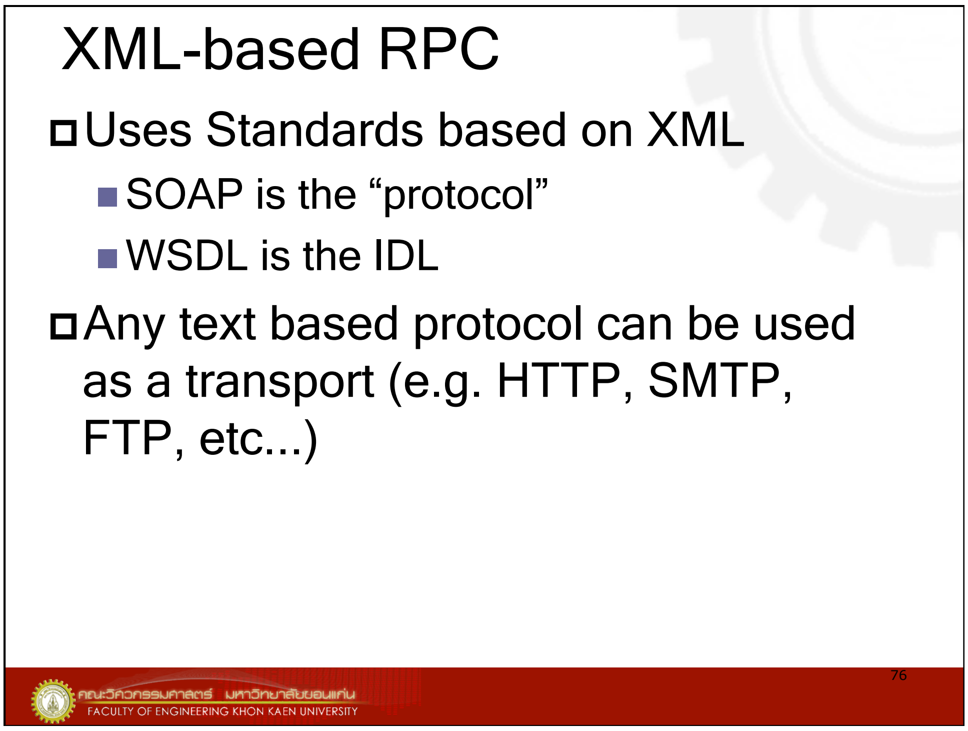
Agenda

- What and Why WSDL?
- Example WSDL Document
- WSDL Document Elements
  - Binding and Extensibility
- Importing & Authoring style
- Application Design & Tools



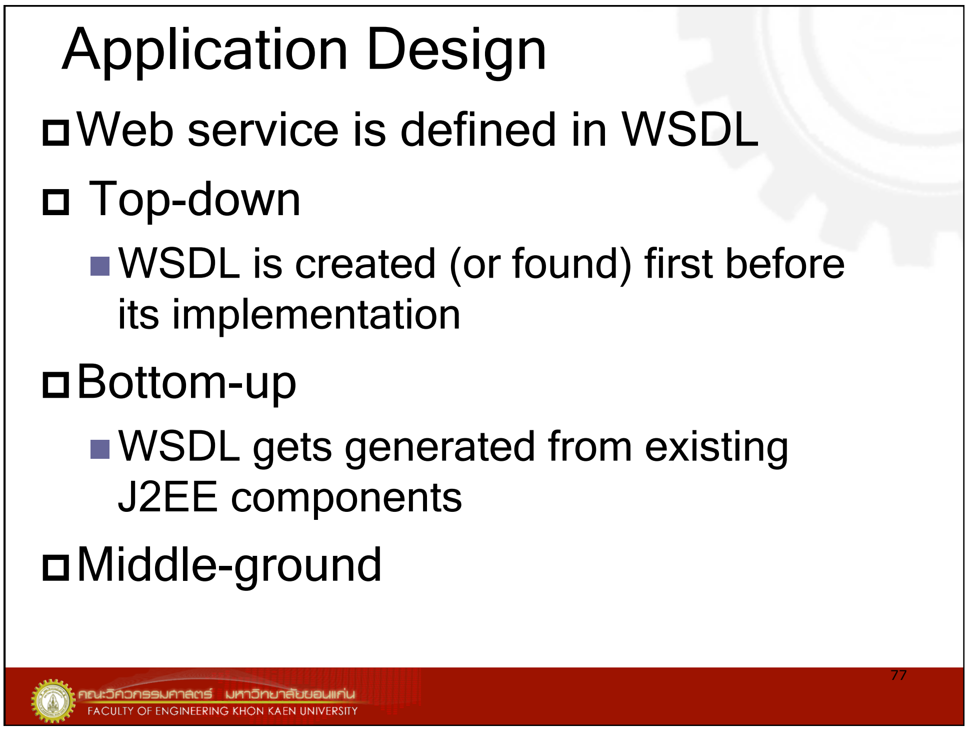
IDL

- Interface Description Language
- Describes programming interfaces in a language neutral way
- Used by tools to statically generate or dynamically configure interfaces, proxies, and ties in a specific environment



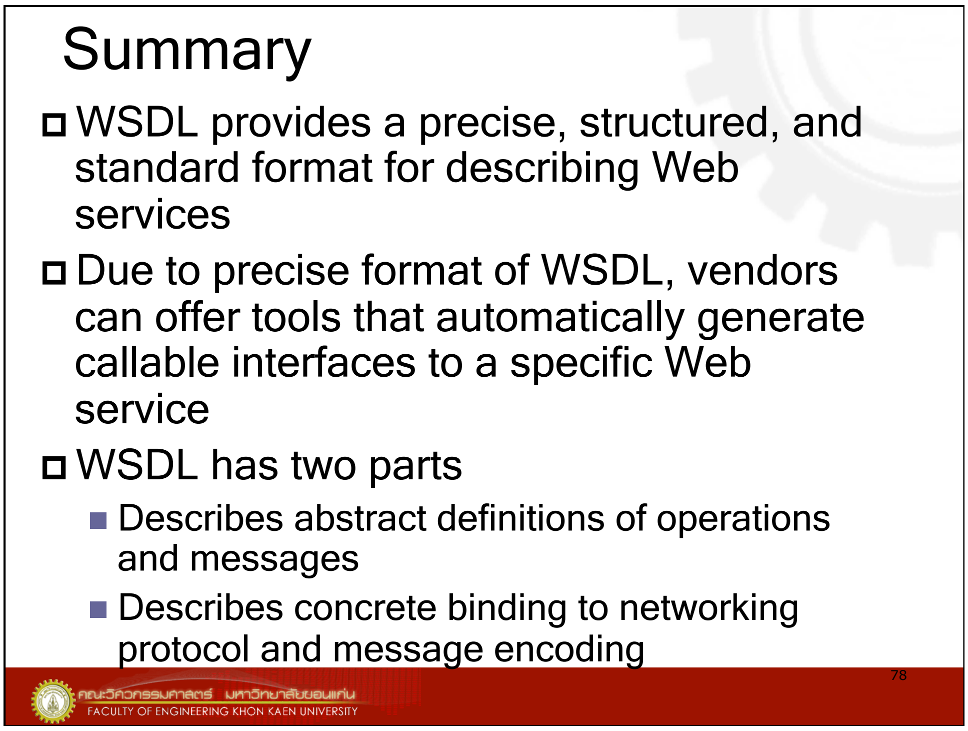
XML-based RPC

- Uses Standards based on XML
  - SOAP is the “protocol”
  - WSDL is the IDL
- Any text based protocol can be used as a transport (e.g. HTTP, SMTP, FTP, etc...)



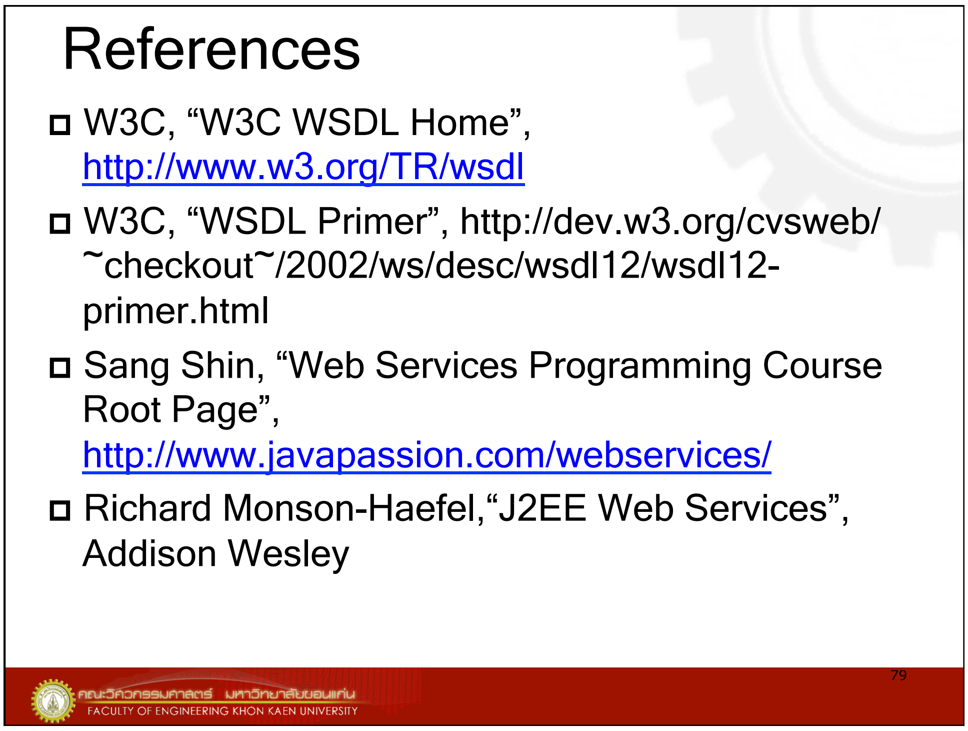
Application Design

- Web service is defined in WSDL
- Top-down
  - WSDL is created (or found) first before its implementation
- Bottom-up
  - WSDL gets generated from existing J2EE components
- Middle-ground



Summary

- WSDL provides a precise, structured, and standard format for describing Web services
- Due to precise format of WSDL, vendors can offer tools that automatically generate callable interfaces to a specific Web service
- WSDL has two parts
  - Describes abstract definitions of operations and messages
  - Describes concrete binding to networking protocol and message encoding



References

- W3C, “W3C WSDL Home”, http://www.w3.org/TR/wsdl
- W3C, “WSDL Primer”, http://dev.w3.org/cvsweb/~checkout~/2002/ws/desc/wsdl12/wsdl12-primer.html
- Sang Shin, “Web Services Programming Course Root Page”, http://www.javapassion.com/webservices/
- Richard Monson-Haefel,“J2EE Web Services”, Addison Wesley

79

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY