UNIVERSITY of STIRLING

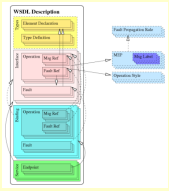
Web Services

WSDL

Web Services Description Language

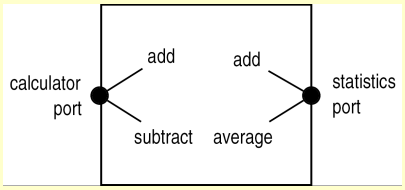
WSDL – Overview of lecture

- Describe the purpose and utility of WSDL.
- Understand the models of interaction.
- Describe the structure of a WSDL document.
- Understand the elements of a Web service represented in WSDL.
- Understand how these elements combine to create a description of a Web service.



Web Services

- a **service** supports:
  - **ports** whose interfaces are defined by **port types**
  - **operations** at ports that can take an input, return an output, and cause a fault
  - **messages** that are sent to or by an operation
  - **faults** that indicate failure of the service (not the underlying communications)



- note that
  - different operations with the same name (e.g. add here) may be supported at different ports
  - operation parameters are optional, e.g. an operation may not produce an output, may send an output without an input, or may not cause a fault
  - although a web service may offer multiple ports, each with multiple operations, in practice services have just one port with multiple operations

UNIVERSITY of STIRLING

Calculator class

```
public class Calculator {
    public int add(int i, int j)
            throws ArithmeticException {
        int result = i + j;
        if ((j >= 0 && result >= i) || (j < 0 && result < i))
            return(result);
        else
            throw new ArithmeticException("addition overflow");
}
    public int subtract(int i, int j)
            throws ArithmeticException {
        int result = i - j;
        if ((j >= 0 && result <= i) || (j < 0 && result > i))
            return(result);
        else
            throw new ArithmeticException("subtraction overflow");
    }
}
```

- Equivalent web service look as follows:
  - a Calculator service with a calculator port
  - operations add and subtract for a pair of integers
  - messages for the input and output of these operations
  - fault ArithmeticException for the case where the result of addition or subtraction is too large
- note that Java does not have the explicit equivalent of a port or of messages
- Axis2 (Apache Extensible Interaction System) is able to convert a simple Java class into a web service (POJO)

UNIVERSITY of STIRLING

WSDL

- Web Service Definition Language (XML)
- Namespace is typically *wsdl*
- A WSDL document describes how to interact with a web service in terms of **data types**, **operations** provided and their **parameters**, **protocols** used, **location** of the service
- WSDL deals with **syntax** (how to call operations) and not **semantics** (what operations do), so other information is needed before a service can be fully understood

→ **Contract** between service provider and requestor

→ Described services can be implemented in **any language** & on **any platform**

UNIVERSITY of STIRLING

Slide 5

5

Uses

- Used by application developers as a spec of the web service
  - Helps with development of both web services and web service clients
  - Source code for (parts of) service and client can be generated from WSDL
  - WSDL can also be generated from a web service implementation
- Used by applications to invoke a web service
  - Dynamically generating a call to the web service based on its description
- Published in service registries
  - Aids discovery and use of web services
- WSDL-described web service can be communicated with using any agreed protocol
  - SOAP (most common)
  - SMTP/MIME
  - HTTP/REST (used for simple cases)

UNIVERSITY of STIRLING

Slide 6

6

WSDL messages

- Web services handle messages in one of two basic styles:
  - **document** style means that each message carries an XML document
  - **rpc** style (cf. remote procedure) means that a request message carries the name of the operation to be invoked plus its parameters, and the resulting response message carries the operation result
- Each of these has two encodings:
  - **encoded** means that the types of all values are explicitly stated
  - **literal** means that values are just given literally – types are implicit
- In practice, only the literal styles are used
- Disadvantage of document/literal over rpc/literal: operations cannot be overloaded (different operations cannot have the same parameters)

WSDL documents

- WSDL separates the abstract description of a service interface from how it is actually supported
- The **Abstract Service Interface Definition**
  - Describes what the web service does (**what** operations it offers)
- The **Concrete Service Implementation**
  - Binds the abstract operations to concrete protocols; **how** to call those operations using those protocols

- Services support one or more **ports** (typically just one)
- Each port supports one or more **operations**
- Each operation may have an **input**, an **output**, and zero or more **faults**

Abstract Service Definition

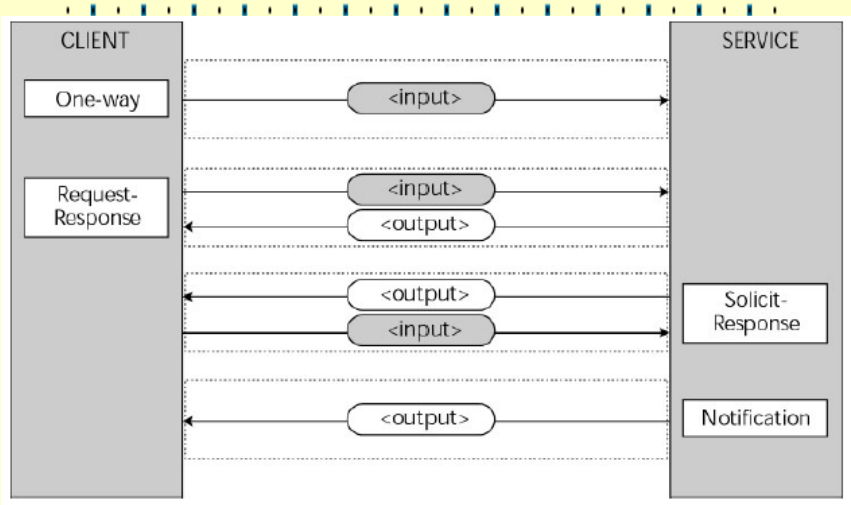
- Data types used by the service
  - Typically XML Schema type definitions
  - Simple types, e.g.: int, float, string, boolean, etc…
  - Complex types, e.g.: Customer, Address, Stock Item
  - Types are used within messages…
- Messages sent/received by the service
  - A message is the payload of a single, one-way communication
  - A message consists of one or more parts
  - Each part is of a certain data type (as defined in data types)
  - Messages are used to make operations…
- How messages combine to form operations
  - At most one input message (input parameters)
  - At most one output message (output parameters)
  - Optional fault descriptions (exceptions)
  - WSDL supports four operation types…
- Operations combine to form a portType

UNIVERSITY of STIRLING

Slide 9

9

Operation Types



"Developing Java Web Services", Figure 5.2

UNIVERSITY of STIRLING

Slide 10

10

Port Types

- Operations are combined to a portType
  - describes the interface(s) of a Web service
  - represent a logical aggregation of operations

```
<definitions>

 <types>
    data type definitions.....
 </types>

 <message>
    definition of the data being communicated....
 </message>

 <portType>
    set of operations.....
 </portType>

 <binding>
    protocol and data format specification....
 </binding>

</definitions>
```

UNIVERSITY of STIRLING

Slide 11

11

Concrete Service Implementation

- Concrete **bindings** of the abstract service interface definition
  - Describe an implementation of a portType
  - Input, output and fault messages in the ops of each port type are mapped to:
    - The transport protocol(s) used
    - The message style (document or rpc)
    - The data encoding style (encoded or literal)
  - Although binding information has to be repeated for each port and operation parameter, these are usually all the same
- Overall Service is defined
  - Name of service
  - Each port has a name, binding and a location
- The entire Web Service is exposed via one or more **ports** (end point)
  - Each binding corresponds to a single port
  - A port is the **actual address** where the service can be found, eg:
    - http://some.web/service if binding to HTTP
    - some.web@service.com if binding to SMTP

UNIVERSITY of STIRLING

Slide 12

12

Tools for WSDL

- all packages for web services include support for WSDL
- Apache Axis2 (Apache Extensible Interaction System) supports:
  - parsing WSDL, and interpreting SOAP messages in the context of this
  - *WSDL2Java* converts from WSDL to Java, creating stubs (outline client code) and skeletons (outline server code)
  - conversely, *Java2WSDL* converts from Java to WSDL
  - basic XML types have a direct Java mapping (e.g. boolean, double, float, int)
  - more complex XML types map onto Java classes

Limitations of WSDL

- Unable to describe complex business processes
  - E.g. sequences of related messages
- Does not describe business level requirements of the service
  - E.g. Quality of Service, Security

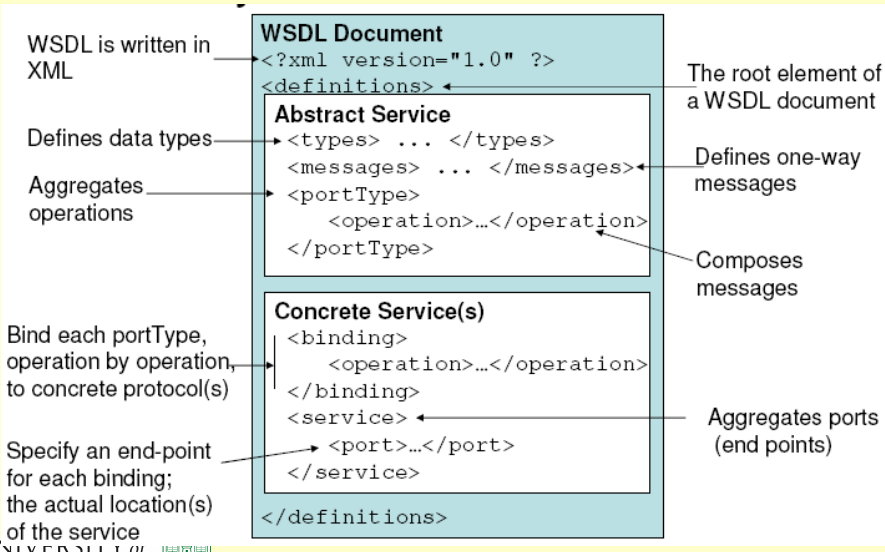
Important Namespaces

- The principle WSDL namespaces are:
  - http://schemas.xmlsoap.org/wsdl/ (Version 1.1)
  - http://www.w3.org/ns/wsdl (Version 2.0)
- The namespace for binding to SOAP messaging
  - http://schemas.xmlsoap.org/wsdl/soap/
- XML Schema –for XML Schema data type encoding
  - http://www.w3c.org/2001/XMLSchema
- SOAP Encoding – for SOAP messages using SOAP encoding
  - http://schemas.xmlsoap.org/soap/encoding/
- WSDL documents may also reference other namespaces
  - Usually this will be for application specific purposes

UNIVERSITY of STIRLING

15

Overview of WSDL documents



UNIVERSITY of STIRLING

16

Example Messages and Porttypes

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>
```

UNIVERSITY of STIRLING

Slide 17

17

Example Binding

```
<binding type="glossaryTerms" name="b1">
   <soap:binding style="document"
   transport="http://schemas.xmlsoap.org/soap/http" />
   <operation>
     <soap:operation soapAction="http://example.com/getTerm"/>
     <input><soap:body use="literal"/></input>
     <output><soap:body use="literal"/></output>
  </operation>
</binding>
```

UNIVERSITY of STIRLING

Slide 18

18

Example: Hello World (Lab 1)

- WSDL for Hello World Lab Example
- Abstract service definition
- One possible binding to SOAP

UNIVERSITY of STIRLING

19

Definitions

- <definitions> is the root element of a WSDL document

```
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:ns="http://ws.apache.org/axis2"
    xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
    xmlns:ns1="http://org.apache.axis2/xsd"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    targetNamespace="http://ws.apache.org/axis2">
    <wsdl:documentation>This is a first Hello World Service.
</wsdl:documentation>
```

- This document also uses elements defined in a number of other Namespaces, e.g.
  - WSDL (wsdl)
  - Binding WSDL to SOAP (soap)
  - XML Schema (xs)

UNIVERSITY of STIRLING

20

WSDL Types

```
<wsdl:types>
        <xs:schema attributeFormDefault="qualified"
                elementFormDefault="qualified" targetNamespace="http://ws.apache.org/axis2">
        <xs:element name="echoMsg">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element minOccurs="0" name="arg" nillable="true"
                                                type="xs:string" />
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
        <xs:element name="echoMsgResponse">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element minOccurs="0" name="return" nillable="true"
                                                type="xs:string" />
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
        </xs:schema>
</wsdl:types>
```

UNIVERSITY of STIRLING

Slide 21

21

WSDL Messages

```
<wsdl:message name="echoMsgRequest">
        <wsdl:part name="parameters" element="ns:echoMsg" />
</wsdl:message>
<wsdl:message name="echoMsgResponse">
        <wsdl:part name="parameters" element="ns:echoMsgResponse" />
</wsdl:message>
```

- Two messages are defined:
  - **echoMsgRequest** is of type "**ns:echoMsg**"
  - **echoMsgResponse** is of type "**ns:echoMsgResponse**"

UNIVERSITY of STIRLING

Slide 22

22

WSDL operations & portType

```
<wsdl:portType name="helloWorldWSPortType">
    <wsdl:operation name="echoMsg">
        <wsdl:input message="ns:echoMsgRequest" wsaw:Action="urn:echoMsg" />
        <wsdl:output message="ns:echoMsgResponse“
                                    wsaw:Action="urn:echoMsgResponse" />
    </wsdl:operation>
</wsdl:portType>
```

- A portType is declared, it is called “**helloWorldWSPortType**”
- The portType has only one operation, called “**echoMsg**”
- This operation consists of:
  - An input which is in the form of the message “**ns:echoMsgRequest**”
  - An output which is in the form of the message “**ns:echoMsgResponse**”

UNIVERSITY of STIRLING

Slide 23

23

WSDL: Binding SOAP11

```
<wsdl:binding name="helloWorldWSSoap11Binding" type="ns:helloWorldWSPortType">
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
                style="document" />
        <wsdl:operation name="echoMsg">
                <soap:operation soapAction="urn:echoMsg" style="document" />
                <wsdl:input>
                        <soap:body use="literal" />
                </wsdl:input>
                <wsdl:output>
                        <soap:body use="literal" />
                </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
```

- A binding called “**helloWorldWSSoap11Binding**” is declared.
- This binds the portType “**ns:helloWorldWSPortType**” to SOAP11 messaging using the document style and literal data type encoding.
- Each input/output (and fault) of each operation in the portType is bound.
- Two more bindings for SOAP12 (identical to SOAP11 binding, but different namespace) and HTTP

UNIVERSITY of STIRLING

Slide 24

24

WSDL: Binding HTTP

```
<wsdl:binding name="helloWorldWSHttpBinding" type="ns:helloWorldWSPortType">
        <http:binding verb="POST" />
        <wsdl:operation name="echoMsg">
                <http:operation location="echoMsg" />
                <wsdl:input>
                        <mime:content type="application/xml" part="parameters" />
                </wsdl:input>
                <wsdl:output>
                        <mime:content type="application/xml" part="parameters" />
                </wsdl:output>
        </wsdl:operation>
</wsdl:binding>
```

UNIVERSITY of STIRLING

WSDL: Service

```
<wsdl:service name="helloWorldWS">
    <wsdl:port name="helloWorldWSHttpSoap11Endpoint“ binding="ns:helloWorldWSSoap11Binding">
       <soap:address
       location="http://127.0.0.1:8080/axis2/services/helloWorldWS.helloWorldWSHttpSoap11Endpoint/"/>
    </wsdl:port>
    <wsdl:port name="helloWorldWSHttpSoap12Endpoint" binding="ns:helloWorldWSSoap12Binding">
      <soap12:address
       location="http://127.0.0.1:8080/axis2/services/helloWorldWS.helloWorldWSHttpSoap12Endpoint/"/>
    </wsdl:port>
    <wsdl:port name="helloWorldWSHttpEndpoint" binding="ns:helloWorldWSHttpBinding">
       <http:address
            location="http://127.0.0.1:8080/axis2/services/helloWorldWS.helloWorldWSHttpEndpoint/" />
     </wsdl:port>
</wsdl:service>
```

- A service called “**helloWorldWS**” is declared
- The service consists of three ports corresponding to the three bindings

UNIVERSITY of STIRLING