Institute of
Public Administration

COMP41530 - Web Services in Cloud Computing

Barry Corish
Associate Lecturer, IPA

Lecture 05

Institute of Public Administration | 57-61 Lansdowne Road | Dublin 4 | Ireland | Ph. +353 1 2403600 | www.ipa.ie

Overview

- Review
- SOAP
- WSDL

Overview

- **Review**
- SOAP
- WSDL

XML Review

- XML is a neutral standard for holding data
  - Language and vendor independent
- On it's own, XML does nothing…
  - Nothing!
  - A set of formats for holding, storing and transporting data.

XML Review

- Readable by both Humans and Machines
- Simple and flexible
- Ideal for use over Internet
- Open Standard
- Unicode support
- Widespread adoption and support

Why is it relevant to us?

- SOA
  - Vendor neutral
  - Widespread support
  - Layer of abstraction
  - Ideal for use over networks
- WebServices
  - This is the data format used for WebServices

Schemas: Defining rules for XML Documents

- XML is very flexible
- We need to be able to define/limit what is acceptable in a given XML Structure
- The set of rules defining what is acceptable is the “schema”.

Defining schemas

- We use XSD (XML Schema Definition Language) to define a schema
- XSD is written in XML
- XML should “validate against” the relevant schema(s)
- DTD (predecessor to XSD) still exists and is widely used

Pre-written schemas

- Agreed standards for format and syntax of XML data
- Hundreds defined and freely available across many industries.
- Before writing your own, do some research
  - See if something suitable is already available
  - The more people using a schema, the more useful it tends to be

Questions?

Overview

- Review
- **SOAP**
- WSDL

What have we got so far?

- XML gives us good, flexible data structures.
  - We can use XML as a container for out data as we store and pass it around
  - We can restrict the structure of the XML document using a Schema
- So, we choose/modify/build a schema, and use it?

Why not?

- If for a simple point to point connections, we can do that.
- …but, we also need to define messages for:
  - XML schema for the message going in
  - XML schema for the response.
    - Even if the response is just "OK, got your message"
  - XML schema for errors

There's already a schema…

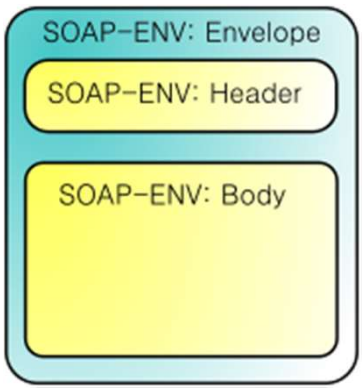
- …for formatting messages
- …for formatting replies
- …for formatting errors
- Not the only choice
  - …but widely used and supported
  - …widely supported by tools (e.g. Eclipse)

SOAP

- **S**imple **O**bject **A**ccess **P**rotocol
- Inter-system communication protocol for messages
- XML based, normally carried over HTTP
  - so ideal for Internet
- Vendor, platform and application neutral
- Standard for Enterprise level WebServices
  - …though this is changing in some areas

SOAP Envelope

- Each message is put into an "Envelope".



SOAP Envelope Definition

```
<?xml version="1.0"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2001/12/soap-
  envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap
  -encoding">
  <soap:Header>
      …header stuff…
  </soap:Header>
  <soap:Body>
      …body stuff..
  </soap:Body>
</soap:Envelope>
```


What goes in the SOAP header?

- Information about the message.
  - Date/Time stamps
  - Security information
  - Addressing/routing information
- NB: Not mandatory to have a header.

What goes in the SOAP body?

- The message!
- A piece of XML
- Two basic ways of “sending” data that you’ll hear about:
  - Document Style – send the whole “document”
  - RPC Literal Style – send an instruction to Create/Read/Update/Delete some date.

Classic example: Stock Price Query

```
<?xml version="1.0"?>
<soap:Envelope
   xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
   soap:encodingStyle="http://www.w3.org/2001/12/soap-
   encoding">

   <soap:Body>
        <m:RequestStockPrice
        xmlns:m="http://www.ise.ie/StockPriceLookup">
               <m:StockCode>IBM</m:StockCode>
        </m:RequestStockPrice>
   </soap:Body>
</soap:Envelope>
```

Classic example: Stock Price Query with Header

```
<?xml version="1.0"?>
<soap:Envelope
   xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
   soap:encodingStyle="http://www.w3.org/2001/12/soap-
   encoding">
   <soap:Header>
        <UsernameToken
   xmlns="http://www.ise.ie/webservices">dave</UsernameToke
   n>
        <PasswordText
   xmlns="http://www.ise.ie/webservices">hello123</Password
   Text>
   </soap:Header>
   <soap:Body>
        …as before..
   </soap:Body>
</soap:Envelope>
```


Classic example: Stock Price Response

```
<?xml version="1.0"?>
<soap:Envelope
   xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
   soap:encodingStyle="http://www.w3.org/2001/12/soap-
   encoding">
   <soap:Body>
        <m:StockPriceResponse
        xmlns:m="http://www.ise.ie/prices">
               <m:Price>45.01</m:Price>
               <m:Currency>EUR</m:Currency>
               <m:Date>2014-10-13</m:Date>
        </m:StockPriceResponse>
   </soap:Body>
</soap:Envelope>
```

Classic example: Stock Price Fault Response

```
<?xml version="1.0"?>
<soap:Envelope
   xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
   soap:encodingStyle="http://www.w3.org/2001/12/soap-
   encoding">

   <soap:Body>
        <soap:fault>
               <faultcode>SOAP-ENV:Client</faultcode>
               <faultstring>No such stock.</faultstring>
        </soap:fault>
   </soap:Body>
</soap:Envelope>
```


Questions?

?

Overview

- Review
- SOAP
- **WSDL**
- Practical 05: Use web services

Before a user can use our WebService…

- They need to know:
  - What information it needs, and in what format
  - What information it will return, and in what format
  - What it does
  - Where it is located
  - How to call it
- We need to describe all these aspects of our WebServices

Why describe a WebService?

- To allow them to be used without “pre-agreement”
- To allow them to be discovered (qf. UDDI)
- To tell a potential user how to use them
- These are all important parts of “loose coupling”

Won’t the XML schema do?

- Only partially
- Schema describes data formats, structures and restrictions
- Doesn’t cover:
  - Where the web service is located
  - What the WebService does
  - How to communicate with the WebService

WSDL

- **W**eb **S**ervices **D**escription **L**anguage
- Describes:
  - What a service does
  - Where to find the service
  - How to use the service
- Written by the WebService provider
- Is the "contract" around the WebService

WSDL

- Written in XML
  - One big XML document (except when nested!)
- Based on a number of Schemas
- Published "beside" the WebService
- "Anyone" can read the WSDL and find out all about the service
  - Only the external interfaces of the service
  - Internal implementation is none of their business

WSDL consists of two parts

- Abstract: service interface definition
  - The operations it supports
  - The data type it understands
- Concrete: service implementation detail
  - Where it is located
  - How to call it
- These are separately defined
- A single operation may have several service implementations

WSDL is generated!

- WSDL is complicated
- WSDL 1.1 is the most common currently in use
- Complicated!
  - Rare to write WSDL by hand
  - Normally generated for you by application software
  - Normally read for you by development environment tools
- WSDL 2.0 under development – much simpler, but not yet so widely used

Major Components of WSDL

- Types
- Messages
- Operations
- portType
- Binding

Major Components of WSDL

- **Types**
- Messages
- Operations
- portType
- Binding

WSDL Types

- Abstract data structures
- Used elsewhere in the WSDL
- Defined in XSD Language
- Think of as “Data Types”, exactly what we defined in the XML/XSD examples
- WSDL starts by defining “Types” (data types)
  - e.g. AlbumType, TrackType
  - Defined in standard XSD

Major Components of WSDL

- Types
  - data types, as per XML/XSD
- **Messages**
- Operations
- portType
- Binding

WSDL Messages

- Defines “there is a message called <X>”
- Describes the “Type” of each message
  - e.g. An “Add Album” message has type “AlbumType”
  - Generally defined in pairs:
    - “Input” message, sent to the WebService
    - “Output” message – the reply that can be sent back by the WebService

WSDL Message Definition

```
<wsdl:message name="AddAlbumMessage">
  <wsdl:part name="AlbumToAdd"
    element="tns:Album"/>
</wsdl:message>
<wsdl:message name="AddAlbumFault">
  <wsdl:part name="AlbumFailedToAdd"
    element="tns:Failure"/>
</wsdl:message>
```

Major Components of WSDL

- Types
  - data types, as per XML/XSD
- Messages
  - has name, and data type
- **Operations**
- portType
- Binding

WSDL Operations

- Defines the operations the WebService can do
- e.g. Add a new album to collection, query a track etc.
- Described in terms of the “Messages”:
  - At most one input message
  - At most one output message
  - Unlimited number of fault messages

WSDL Operation Models

- One-way (occasional)
  - The operation can receive a message but will not return a response
- Request-response (>95% of the time)
  - The operation can receive a request and will return a response
- Solicit-response (not typical)
  - The operation can send a request and will wait for a response
- Notification (not typical)
  - The operation can send a message but will not wait for a response

WSDL Request/Response Operation Example

```
<wsdl:operation name="AddAlbum">
  <wsdl:input name="AddThisAlbum"
   message="AddAlbumMessage"/>
  <wsdl:output name="AlbumAdded"
   message="AddAlbumResponse"/>
  <wsdl:fault name="AddAlbumFault"
   message="AddAlbumFault"/>
</wsdl:operation>
```

WSDL One Way Operation Example

```
<wsdl:operation name="AddAlbum">
  <wsdl:input name="AddThisAlbum"
   message="AddAlbumMessage"/>
</wsdl:operation>
```


Major Components of WSDL

- Types (data types, as per XML/XSD)
- Messages (has name, and data type)
- Operations (a thing you can do, and the messages involved)
- **portType**
- Binding

WSDL portType

- Logical groups of operations
  - Not the same as TCP ports!
- Structured around business operation
- Typically, one Port per WSDL document
- e.g The portType “AlbumOperationsPortType” contains operations:
  - AddAlbum
  - QueryAlbum
  - ModifyAlbum
  - RemoveAlbum

WSDL portType example

```
<wsdl:portType name =“AlbumOperationsPortType”>
    <wsdl:operation name=“AddAlbum“>
        <wsdl:input name=“AddThisAlbum“
          message=“AddAlbumMessage"/>
        <wsdl:output name=“AlbumAdded“
          message=“AddAlbumResponse"/>
    </wsdl:operation>
    <wsdl:operation name=“QueryAlbum“>
        <wsdl:input name=“AlbumQuery“
          message=“AlbumQueryMessage"/>
        <wsdl:output name=“AlbumDetails“
          message=“AlbumDetailsMessage"/>
    </wsdl:operation>
    ..etc for other operations…
</wsdl:portType>
```

Major Components of WSDL

- Types
  - data types, as per XML/XSD
- Messages
  - has name, and data type
- Operations
  - a thing you can do, and the messages involved
- portType
  - a group of related operations
- **Binding**

WSDL Bindings

- Take a group of operations (portType)
- Define where to find them
- Define <service> and <port>

WSDL Bindings Example

- Take a group of operations (portType)
- Define where to find them
- Define <service> and <port>
- In our case, define URL at which the portType can be found
- But can also be defined as reachable of SMTP, MIME etc.
  - But in practice, nearly always HTTP or HTTPs

Major Components of WSDL

- Types
  - data types, as per XML/XSD
- Messages
  - has name, and Type as above.
- Operation
  - a thing you can do, and the messages involved
- portType
  - a group of related operations
- Binding
  - where to find a portType (generally a URL), and how to talk to it

Overview

- Review
- SOAP
- WSDL