Web Services and SOA

61FIT3NPR -Network Programming

Faculty of Information Technology
Hanoi University
Fall 2020

Learning Objectives

- To understand the basics of Web services and SOA
- To understand potential applications of Web services and SOA in e-business and enterprise computing, in particular, for business process integration
- To know the some technological details of SOA: UDDI, WSDL, and SOAP

New Age of Distributed Computing

- Convergence of two technologies
- The Web:
  - Universal communication
  - HTTP, XML
- Service-oriented computing:
  - Exposing data and business logic through a programmable interface
  - EJB, RPC, RMI, CORBA, DCOM
  - SOAP, RESTFul

What is SOA?

- Contemporary Service-Oriented Architectures (SOA) represents an architecture that promotes service-orientation through the use of Web services.
- All functions, or services, are defined using a description language and have invokable interface that are called to perform business processes.

What is a Web Service?

- A Web service is a software system designed to support interoperable machine-to-machine interaction over a network.
- It has an interface described in a machine-processable format (specifically WSDL).
- Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards."
- http://www.w3.org/TR/ws-arch/

Key features of Web Services

- A modular, well-defined, encapsulated function
- Used for loosely coupled integration between applications or systems
- Based on XML, transported in two forms:
  - Synchronous (RPC)
  - Asynchronous (messaging)
  - Both over Simple Object Access Protocol (SOAP)
- Specified in *Web Services Description Language (WSDL)*
- Sometimes advertised and discovered in a service registry – *Universal Description, Discovery and Integration (UDDI)*
- Over Intranet and Internet

Use of SOA and Web Services

- Facilitates:
  - Marketing efforts
  - E-Commerce
  - Personalization
  - Direct services to end users
- Strategies:
  - Focus now on partnerships
  - Integration
  - Direct communication
  - Automating processes across organizational boundaries

Distributed Computing Technologies

- **CORBA (OMG)**
  It is standards-based, vendor-neutral, and language-agnostic. Very powerful but limited however by its complicated way of utilizing the power and flexibility of the Internet.
- **DCOM (Microsoft)**
  Distributed Computing platform closely tied to Microsoft component efforts such as OLE, COM and ActiveX.
- **RMI (Sun Microsystems)**
  Java based effort which doesn't play well with other languages. The J2EE platform integrated RMI with IIOP.
- **Web Services (W3C)**
  Web services are more of an evolution than a revolution

What is a Web Service?

Definition

A **Web Service** is a standards-based, language-agnostic software entity, that accepts specially formatted requests from other software entities on remote machines via vendor and transport neutral communication protocols, producing application specific responses.

- **Standards based**
- **Language agnostic**
- **Formatted requests**
- **Remote machines**
- **Vendor neutral**
- **Transport neutral**
- **Application specific responses**

Benefits of Web Services

- **Loosely Coupled**
  Each service exists independently of the other services that make up the application. Individual pieces of the application to be modified without impacting unrelated areas.
- **Ease of Integration**
  Data is isolated between applications creating 'silos'. Web Services act as glue between these and enable easier communications within and across organisations.
- **Service Reuse**
  Takes code reuse a step further. A specific function within the domain is only ever coded once and used over and over again by consuming applications.

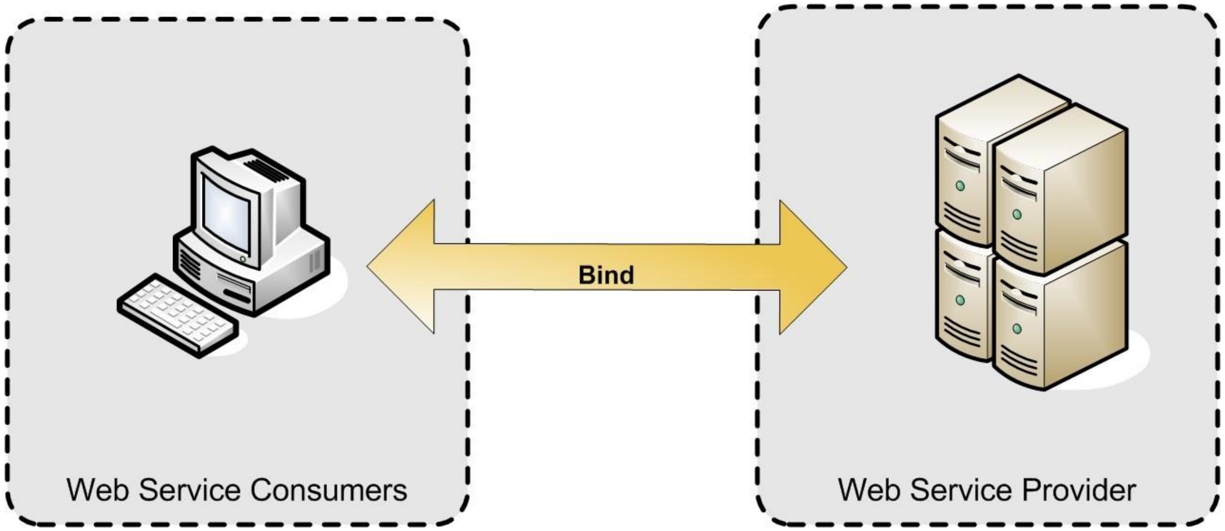
Web Services Architectures

The simplest Web Service System

The simplest Web service system has two participants:

- A service **producer** (provider)
- A service **consumer** (requester).

The provider presents the interface and implementation of the service, and the requester uses the Web service.

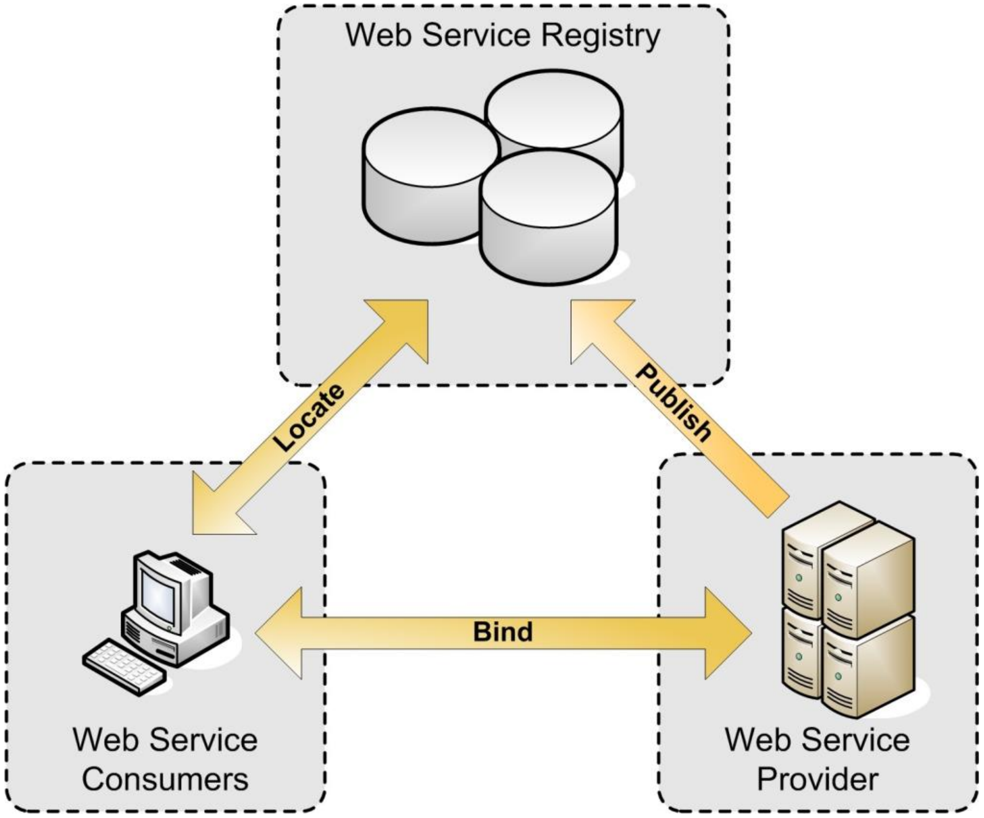


Web Services Architectures

A Service Oriented Architecture (SOA)

A more sophisticated system:

- A **registry**, acts as a broker for Web services.
- A **provider**, can publish services to the registry
- A **consumer**, can then discover services in the registry



Programming Interfaces and APIs

- API = Application Programming Interface
  - ***Don't confuse this with a user Interface***
- Defines a set of functions that may be called by application programs
  - Like a library
  - But application developer may not have access to the code implementing the functions
    - E.g., being an eBay developer means you can use the API, not that you can see how it is implemented
  - And the functions may even be executed on a different computer!
    - E.g., eBay developers are using functions located on eBay servers, not their own computers

Amazon and Google Experiment with Web Services

- Both Google and Amazon have conducted open experiments with Web services.
- Why? To allow partners to develop custom user interfaces and applications that work Google and Amazon data and services.
- You can download their APIs and try them.
  - http://www.google.com/apis/
  - http://www.amazon.com/webservices

What is Google API?

- The Google Web APIs service is a beta web program that developers to easily find and manipulate information on the web.
- Google Web APIs are for developers and researchers interested in using Google as a resource in their applications.
- The Google Web APIs service allows software developers to query more than 3 billion web documents directly from their own computer programs.
- Google uses the SOAP and WSDL standards to act as an interface between the user's program and Google API.
- Programming environments such as Java, Perl, Visual Studio .NET are compatible with Google API.

Definitions from http:// www.google.com/apis/

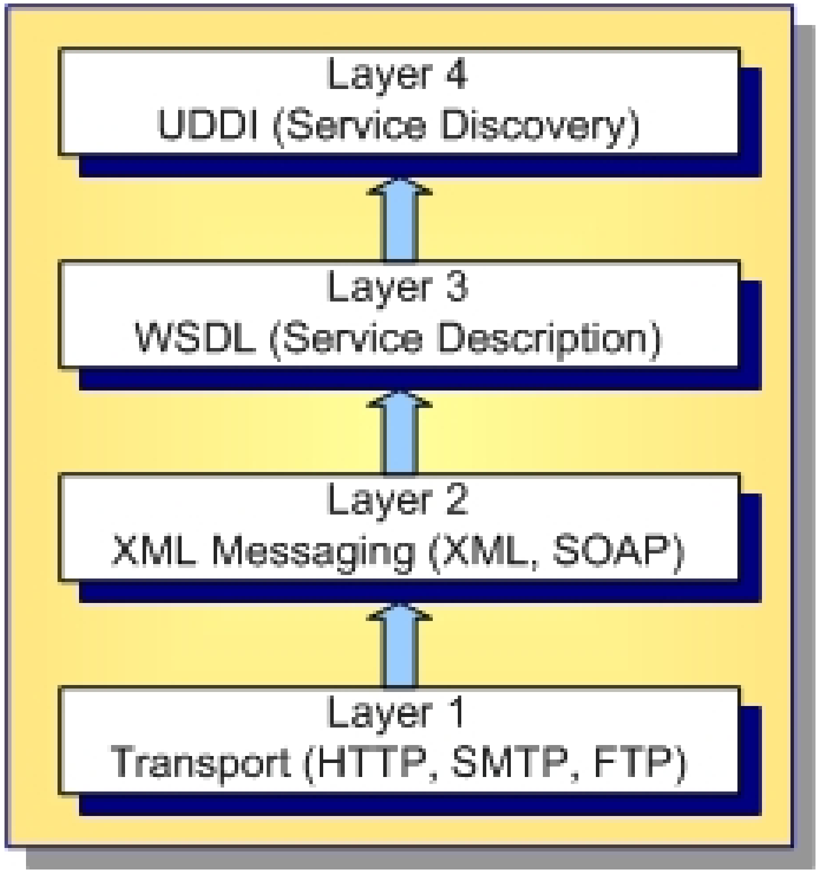
What can you do with the API

- Developers can issue search requests to Google's index of more than 3 billion web pages.
  - and receive results as
    - structured data,
      - Estimated number of results, URL’s, Snippets, Query Time etc.
    - access information in the Google cache,
    - and check the spelling of words.
- Google API

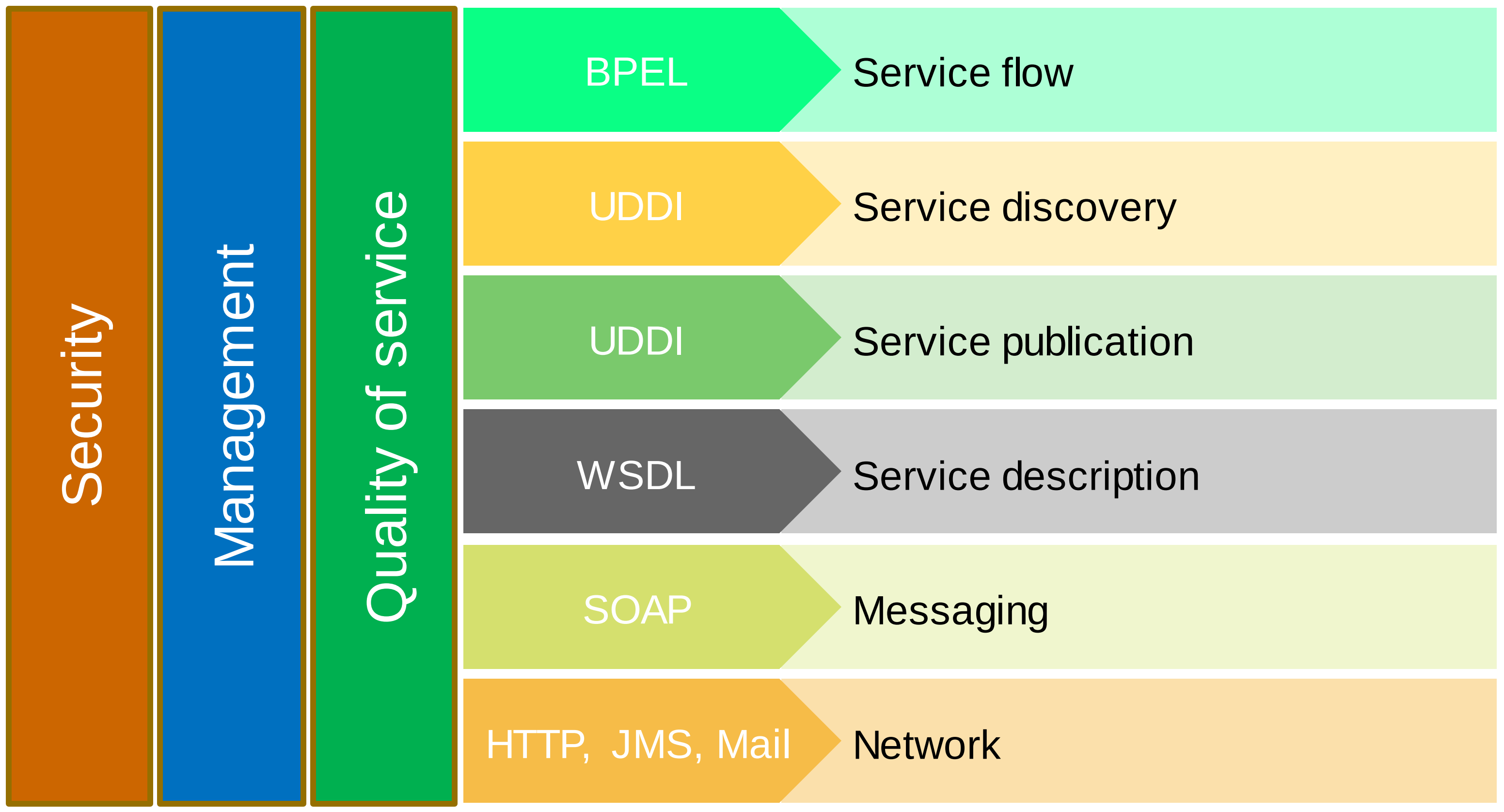
Web Services Protocol Stack

- Understanding the Web Services infrastructure requires that you have some exposure to:
  - Extensible Markup Language (XML)
  - Simple Object Access Protocol (SOAP)
  - Web Services Description Language (WSDL)
  - Universal Description, Discovery, and Integration (UDDI)

Figure 2.3. *Layers of the Web Services Protocol Stack*

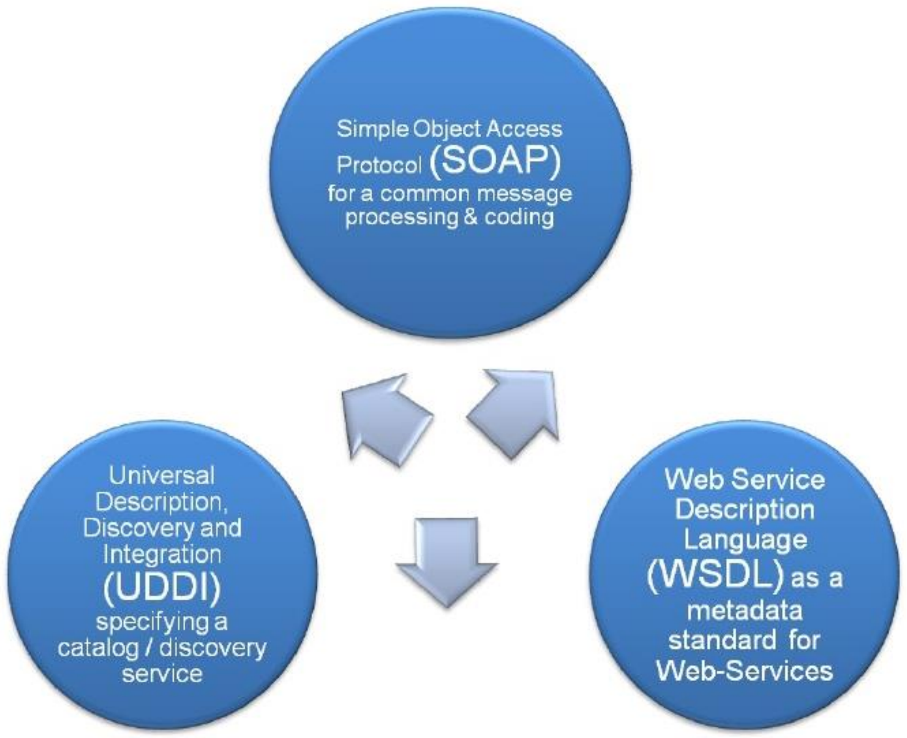

Layer 4
UDDI (Service Discovery)
Layer 3
WSDL (Service Description)
Layer 2
XML Messaging (XML, SOAP)
Layer 1
Transport (HTTP, SMTP, FTP)

Web Services Stack



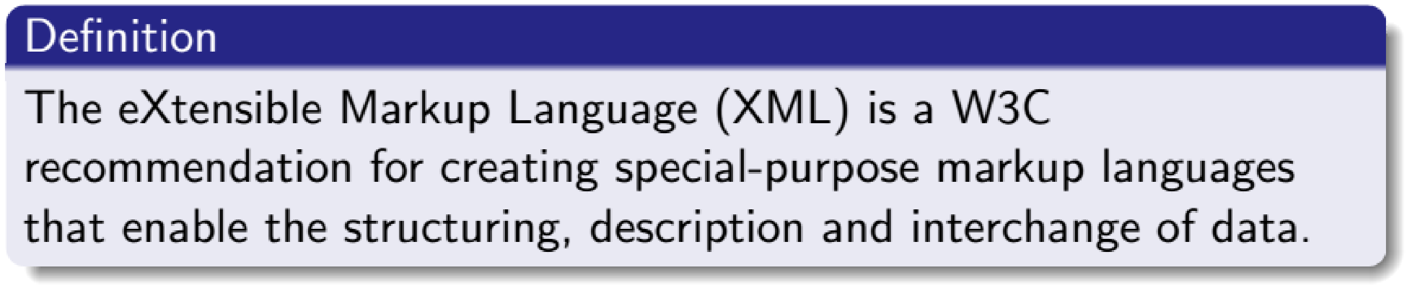
Web Services Specification

There are three basic specifications, forming the basis for state-of-the-art Web-Service technology


Simple Object Access Protocol (SOAP) for a common message processing & coding
Universal Description, Discovery and Integration (UDDI) specifying a catalog / discovery service
Web Service Description Language (WSDL) as a metadata standard for Web-Services

XML

eXtensible Markup Language (XML)



Definition

The eXtensible Markup Language (XML) is a W3C recommendation for creating special-purpose markup languages that enable the structuring, description and interchange of data.

- A simplified subset of SGML capable of describing many different kinds of data for any imaginable application domain.
- It facilitates the sharing of structured text and information in databases and across the Internet.

eXtensible Markup Language (XML)

- Languages based on XML are themselves described in a formal way, allowing programs to modify and validate documents in these languages without prior knowledge of their form.
- Separate syntax from semantics.
- Inherently supports internationalization (Unicode) and platform independence.

XML Building Blocks

- Element

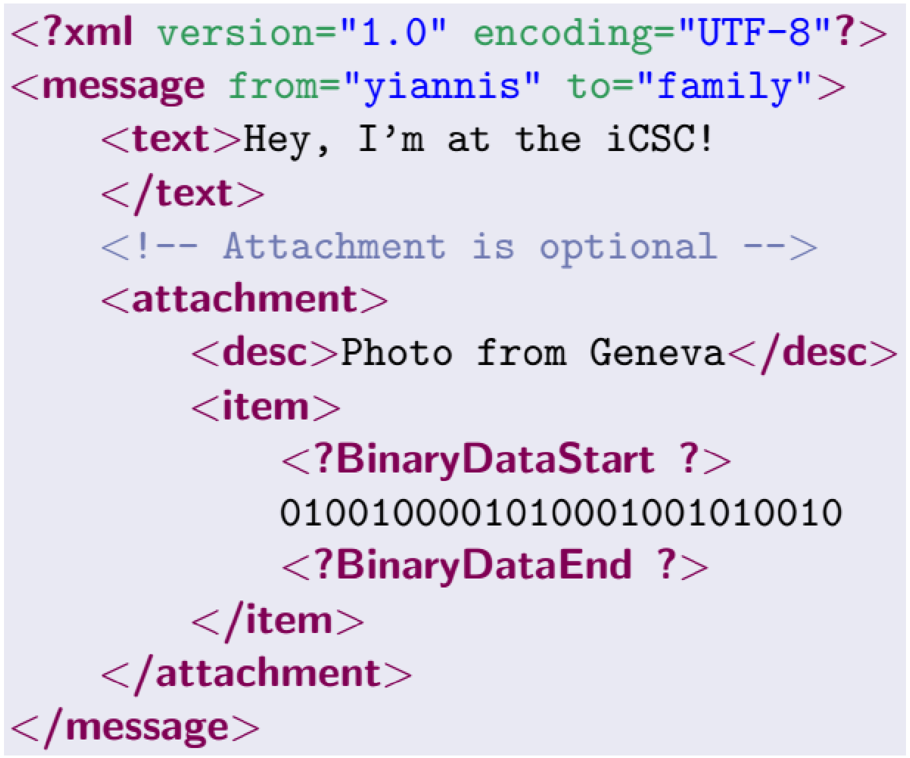
  The pairing of a start tag and an end tag.
- Attribute

  A name-value pair that is part of a starting tag of an Element.
- Processing instruction

  Special directives to the application that will process the XML document.
- Comments

  Messages helping a human reader understand the source code.
- Character data
  - Characters (in a specific encoding)
  - Entities
  - Whitespace

An XML Document



```
<?xml version="1.0" encoding="UTF-8"?>
<message from="yiannis" to="family">
    <text>Hey, I’m at the iCSC!
    </text>
    <!-- Attachment is optional -->
    <attachment>
        <desc>Photo from Geneva</desc>
        <item>
            <?BinaryDataStart ?>
            0100100001010001001010010
            <?BinaryDataEnd ?>
        </item>
    </attachment>
</message>
```

An XML Document consists of:

- Optional prolog
- A root element
- Comments
- Processing Instructions

But…

Some Problems

And how we solved them!

The problems in the previous example relate with the:

- Physical Structure of the document
  Well formedness (Parsers)
- Logical Structure of the document
  Validity (Schemas). Semantics of the elements?
- Element Name clashes between Documents
  Namespaces

XML Namespaces

Motivating the Problem

Solve the problem of recognition and collision of elements in an XML Document.

- Recognition

  How does an XML processing application distinguish between the XML elements that describe the message and the XML elements that are part of a Purchase Order?

- Collision

  Does the element description refer to attachment descriptions in messages or order item descriptions? Does the item element refer to an item of attachment or an order item?

XML Namespaces

Detailing the Solution

XML Namespaces uses Uniform Resource Identifiers for uniquely qualifying local names. As URIs can be long and typically contain
characters that arent allowed in XML element names, the process of including namespaces in XML document involved two steps:

- A namespace identifier is associated with a prefix, a name that contains only legal XML element name characters with the exception of the colon (;)
- Qualified names are obtained as a combination of the prefix, the colon character, and the local element name
  `Qualified Name = Namespace Identifier + Local Name`

A Namespaces XML Document

```xml
<msg:message from="yiannis" to="family"
  xmlns:msg="http://www.w2c.com/ns/email"
  xmlns:po="http://www.w2c.com/ns/purchase">
   <msg:text>
      <msg:desc>A Purchase Order</msg:desc>
      <msg:item>
         <po:order>
            <po:item>
               <po:desc>Laptop Computer</po:desc>
               <po:price>1300 GBP</po:price>
            </po:item>
         </po:order>
      </msg:item>
   </msg:text>
</msg:message>
```

XML Namespaces

A couple more last things

- Default Namespaces

  Adding a prefix to every element in the document decreases readability and increases document size. Therefore, XML Namespaces allow us to use a default namespace in a document. Elements belonging to the default namespace don't require prefixes.
- Namespace prefixed attributes

  Attributes can also have namespaces associated with them. The desire to extend the information provided by an XML element without having to make changes directly to its document type.

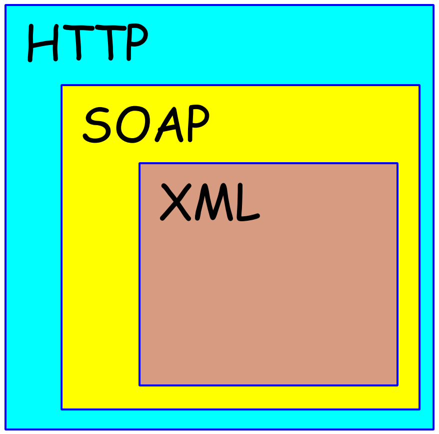
XML Schema

- An XML Schema enables the following:
  - Identification of the elements that can be in a document
  - Identification of the order and relation between elements
  - Identification of the attributes of every element and whether they’re optional or required or have some other special properties
  - Identification of the datatype of attribute content
- Think of it as an elaborate UML Class diagram where classes only have field and no methods.

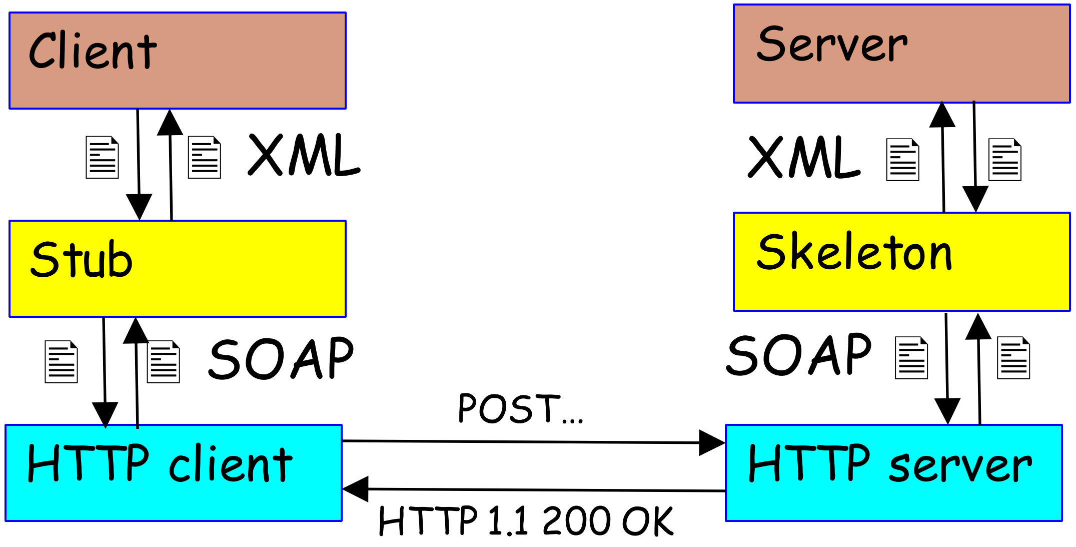
SOAP

Simple Object Access Protocol

SOAP is a technology to support the exchange of XML-coded messages over a transport protocol, such as HTTP and SMTP. (*wire stack*)



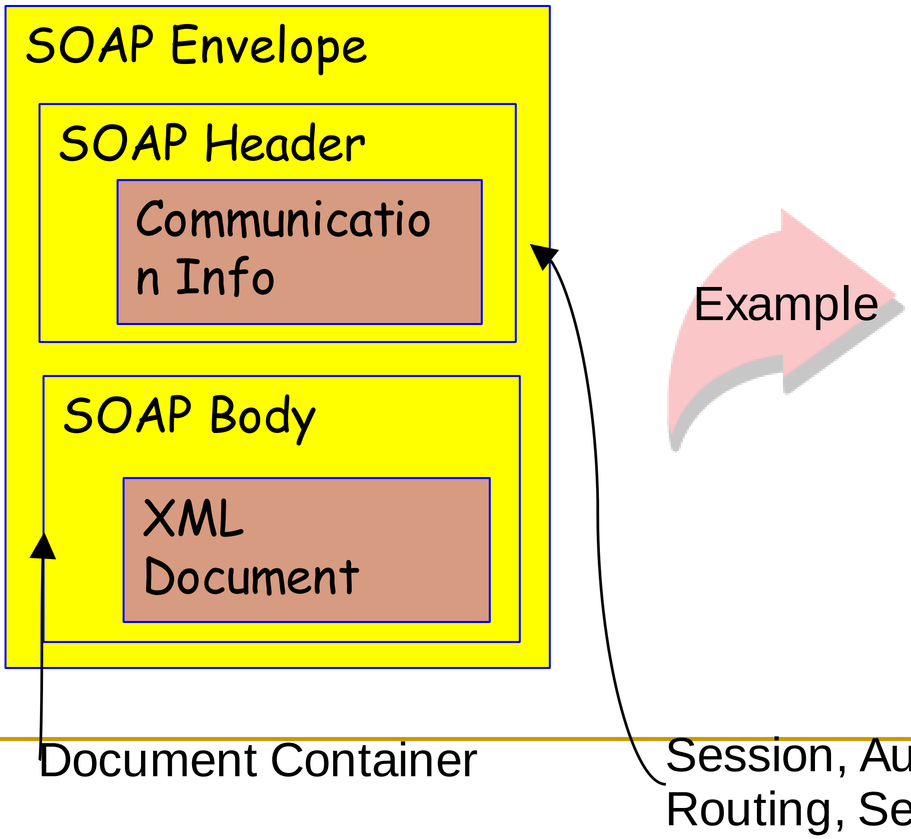
Protocols Folding



SOAP basic mechanism

Simple Object Access Protocol

A SOAP runtime engine basically adds a XML envelope to an existing XML document



```
<soap:Envelope>
  <soap:Header>
    <axis:SessionKey>
      SDHH37TYEW7R7
    </axis:SessionKey>
  </soap:Header>
  <soap:Body>
    <GetPrice>
    <Item>Apples</Item>
    </GetPrice>
  </soap:Body>
</soap:Envelope>
```

SOAP Encoding

Dealing directly with XML messages is not easy.

Therefore, SOAP provides a “RPC emulation” technology

The result: developers work with web services like with traditional RPC (e.g. CORBA,DCOM,DCE)

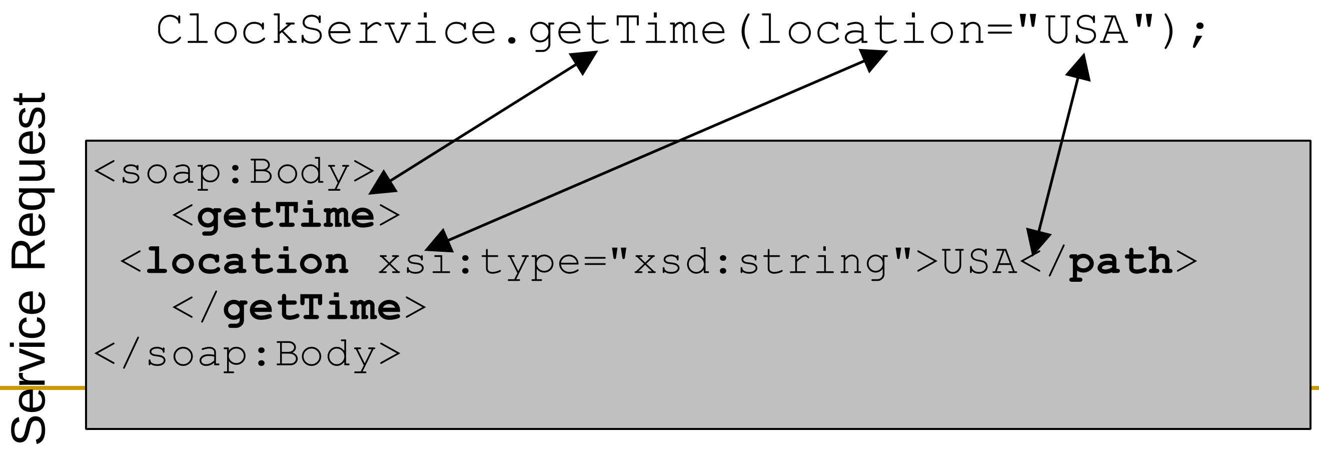
The “RPC emulation” is named SOAP encoding.

Fundamentally it is a set of rules to map a procedure invocation to a XML document.

SOAP Encoding

The rules:

- method name -> first level element in the SOAP Body
- arguments identifiers -> second level elements
- arguments values -> third level elements
- arguments types -> attribute xsi:type



```
ClockService.getTime(location="USA");
```

```
<soap:Body>
   <getTime>
 <location xsi:type="xsd:string">USA</path>
   </getTime>
</soap:Body>
```

WSDL

Web Services Description Language (WSDL)

Web Services Description Language (WSDL) is an XML format for describing all the information needed to invoke and communicate with a Web Service. It gives the answers to the questions Who? What? Where? Why? How? A service description has two major components:

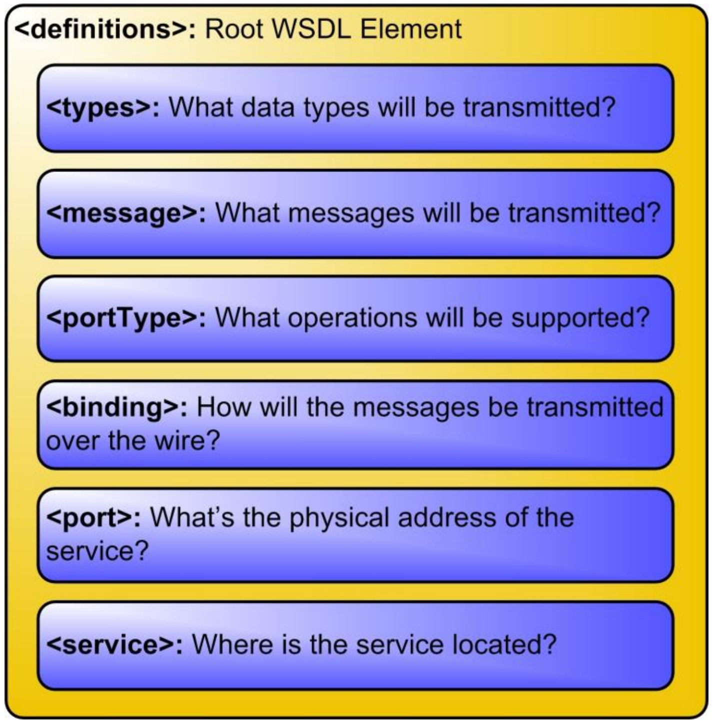
- **Functional Description**
  Defines details of how the Web Service is invoked, where it's invoked. Focuses on the details of the syntax of the message and how to configure the network protocols to deliver the message.
- **Nonfunctional Description**
  Provides other details tha are secondary to the message (such as security policy) but instruct the requestor's runtime environment to include additional SOAP headers.

WSDL Document Structure

The 6 basic building blocks

A WSDL Document is a set of definitions with a single root element. Services can be defined using the following XML elements:

- **Types**, think Data Type
- **Message**, think Methods
- **PortType**, think Interfaces
- **Binding**, think Encoding Scheme
- **Port**, think URL
- **Service**, many URLs


<definitions>: Root WSDL Element

<types>: What data types will be transmitted?

<message>: What messages will be transmitted?

<portType>: What operations will be supported?

<binding>: How will the messages be transmitted over the wire?

<port>: What's the physical address of the service?

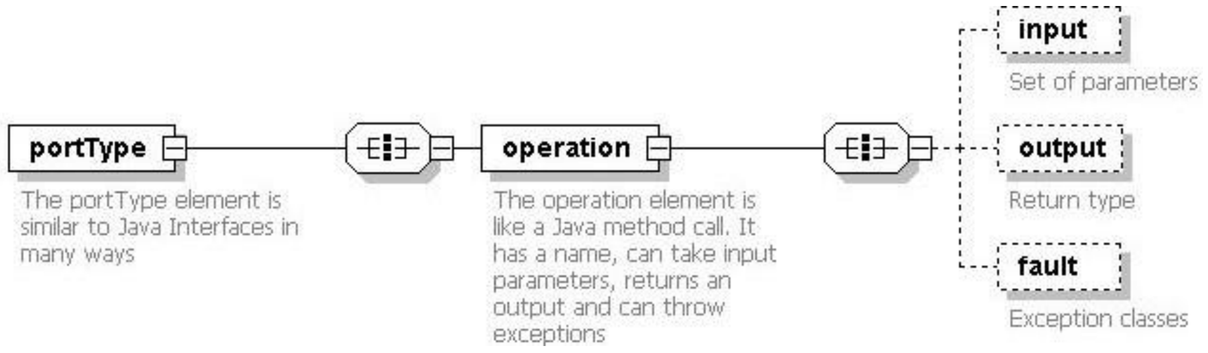
<service>: Where is the service located?

PortType Element
Definition and Usage

Definition

The portType element describes the interface to a Web Service

- A WSDL Document can contain zero or more portType
- A portType element contains a single name attribute. Naming convention *nameOfWebService* PortType
- A portType contains one or more operation elements, with a name attribute can contain input, output and fault elements


portType
The portType element is similar to Java Interfaces in many ways
operation
The operation element is like a Java method call. It has a name, can take input parameters, returns an output and can throw exceptions
input
Set of parameters
output
Return type
fault
Exception classes

PortType Element
Example

Example

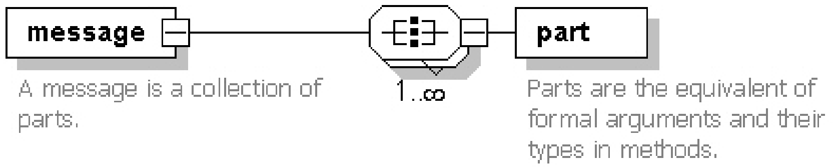
```
<!-- Port Type Definition Example -->
<portType name="weatherCheckPortType">
  <operation name="checkTemperature">
    <input message="checkTemperatureRequest"/>
    <output message="checkTemperatureResponse"/>
  </operation>
  <operation name="checkHumidity">
    <input message="checkHumidityRequest"/>
    <output message="checkHumidityResponse"/>
  </operation>
</portType>
```

Message Element
Definition and Usage

Definition

A message is a collection of parts; intuitively a `part` is a named argument with its type. A message is a collection of these parts.

- A WSDL document can contain zero or more message elements.
- Each message element can be used as an input, output or fault message within an `operation` .
- The type attribute of `part` can be any standard data type from the XSD Schema or a user defined one.


message
A message is a collection of parts.
1..∞
part
Parts are the equivalent of formal arguments and their types in methods.

Message Element
Example

Example

```
<!-- Message Definitions -->
<message name="checkTemperatureRequest">
      <part name="location" type="xsd:string">
</message>
<message name="checkTemperatureResponse">
      <part name="result" type="xsd:double">
</message>
<message name="checkHumidityRequest">
      <part name="location" type="xsd:string">
</message>
<message name="checkHumidityResponse">
      <part name="result" type="ns:HummidityType"
</message>
```

Types Element

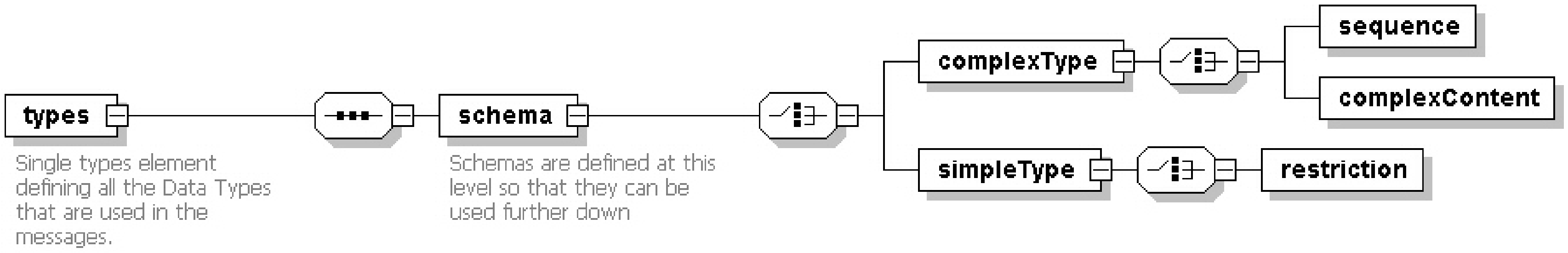
Definition and Usage

Definition

Custom user data types defined in an abstract way.

- The default type system in WSDL is the XML Schema (XSD)
- A WSDL document can have at most one types element.
- The types element can contain simpleType or complexType.
- At the lowest level elements intuitively named (again!) element are defined with a name and a type attribute.

NOTE! The diagram bellow is incomplete! This is considered an advanced topic and for more information you should look at data modelling using the XML Schema.


types
Single types element defining all the Data Types that are used in the messages.
schema
Schemas are defined at this level so that they can be used further down
complexType
sequence
complexContent
simpleType
restriction

Types Element
Example

Example

```
<!-- Type Definitions -->
<types>
   <xsd:schema targetNamespace="http://weather.com/ns"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:complexType name="HumidityType">
         <xsd:sequence>
            <xsd:element name="loc" type="xsd:string">
            <xsd:element name="humd" type="xsd:double">
            <xsd:element name="temp" type="xsd:double">
         </xsd:sequence>
      </xsd:complexType>
   </xsd:schema>
</types>
```

Binding Element
Definition and Usage

Definition
The binding element specifies to the service requester how to format the message in a protocol-specific manner.

- Each portType can have one or more binding elements associated with it.
- For a given portType the binding element has to specify an messaging and transport pair. (SOAP/HTTP, SOAP/SMTP, etc).

Binding Element

```
<binding name="WeatherBinding"  type="weatherCheckPortType">
   <soap:binding style="rpc"
      transport="http://schemas.xmlsoap.org/soap/http"
      / >
   <operation  name="checkTemperature">
      <soap:operation soapAction=""   / >
         <input>
            <soap:body use="encoded"  namespace="checkTemperature"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      / >
         </input>
         <output>
            <soap:body use="encoded"  namespace="checkTemperature"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      / >
         </output>
   </operation>
</binding>
```

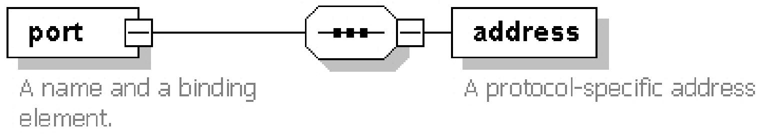
Port Element

Definition, Usage & Example

Definition

The port element specifies the network address of the endpoint hosting the Web Service.

- It associates a single protocol-specific address to an individual binding element.
- Ports are named and must be unique within the document.



Example

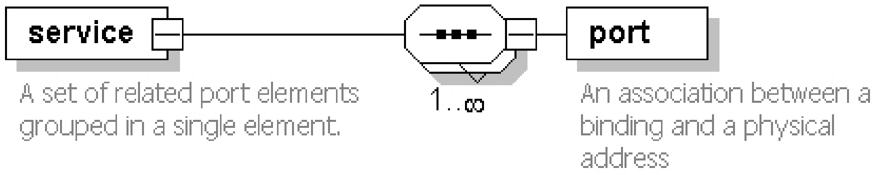
```
<port name="WeatherCheck"
        binding="wc:WeatherCheckSOAPBinding">
   <soap:address location="http://host/WeatherCheck"/>
</port>
```

Service Element
Definition and Usage

Definition

The service element is a collection of related port elements identified by a single service name.

- A WSDL Document is allowed to contain multiple service elements, but conventionally contains a single one.
- Each service must be uniquely named.
- The naming convention is GeneralInfoService


service
A set of related port elements grouped in a single element.
1..∞
port
An association between a binding and a physical address

Service Element
Example

Example

```
<!-- Service definition -->
<service name="WeatherCheckService">
  <port name="WeatherCheckSOAP"
        binding="wc:WeatherCheckSOAPBinding">
  <soap:address location="http://host/WeatherCheck"/>
  </port>
  <port name="WeatherCheckSMTP"
        binding="wc:WeatherCheckSMTPBinding">
  <soap:address location="http://host/WeatherCheck"/>
</port>
</service>
```

Concluding Remarks

In this first lecture we saw

- the position of Web Services within the Distributed Computing Environment.
- the XML primitives and touched upon Namespaces and Schemas.
- how SOAP is used for transferring platform and language independent messages between software entities on different hosts.
- how to describe Web Services using WSDL.

Web Service frameworks (Java)

Apache Axis2

- http://axis.apache.org/axis2/java/core/

Apache CXF

- http://cxf.apache.org

Sun Metro

- https://metro.java.net/

Spring WS

- http://projects.spring.io/spring-ws/