أكاديمية الحكومة الإلكترونية الفلسطينية

The Palestinian eGovernment Academy

www.egovacademy.ps

Tutorial III:

Process Integration and Service Oriented Architectures

Session 7
SOAP

Prepared By

Mohammed Aldasht

Reviewed by

Prof. Marco Ronchetti and Prof. Paolo Bouquet, Trento University, Italy

European Commission

TEMPUS

About

This tutorial is part of the PalGov project, funded by the TEMPUS IV program of the Commission of the European Communities, grant agreement 511159-TEMPUS-1-2010-1-PS-TEMPUS-JPHES. The project website: www.egovacademy.ps

Project Consortium:

- Birzeit University, Palestine (**Coordinator**)
- Palestine Polytechnic University, Palestine
- Palestine Technical University, Palestine
- Ministry of Telecom and IT, Palestine
- Ministry of Interior, Palestine
- Ministry of Local Government, Palestine
- University of Trento, Italy
- Vrije Universiteit Brussel, Belgium
- Université de Savoie, France
- University of Namur, Belgium
- TrueTrust, UK

Coordinator:
Dr. Mustafa Jarrar
Birzeit University, P.O.Box 14- Birzeit, Palestine
Telfax:+972 2 2982935 mjarrar@birzeit.edu

© Copyright Notes

Everyone is encouraged to use this material, or part of it, but should properly cite the project (logo and website), and the author of that part.

No part of this tutorial may be reproduced or modified in any form or by any means, without prior written permission from the project, who have the full copyrights on the material.

Attribution-NonCommercial-ShareAlike
CC-BY-NC-SA

This license lets others remix, tweak, and build upon your work non-commercially, as long as they credit you and license their new creations under the identical terms.

Tutorial Map

Intended Learning Objectives
A: Knowledge and Understanding
3a1: Demonstrate knowledge of the fundamentals of middleware.
3a2: Describe the concept behind web service protocols.
3a3: Explain the concept of service oriented architecture.
3a4: Explain the concept of enterprise service bus.
3a5: Understanding WSDL service interfaces in UDDI.
B: Intellectual Skills
3b1: Design, develop, and deploy applications based on Service Oriented Architecture (SOA).
3b2: use Business Process Execution Language (BPEL).
3b3: using WSDL to describe web services.
C: Professional and Practical Skills
3c1: setup, Invoke, and deploy web services using integrated development environment.
3c2: construct and use REST and SOAP messages for web services communication.
D: General and Transferable Skills
d1: Working with team.
d2: Presenting and defending ideas.
d3: Use of creativity and innovation in problem solving.
d4: Develop communication skills and logical reasoning abilities.

Title	T	Name
Session0: Syllabus and overview	0	Aldasht
Sesson1: Introduction to SOA	2	Aldasht
Session2: XML namespaces & XML schema	2	Aldasht
Session 3: Xpath & Xquery	4	Romi
Session4: REST web services	3	M. Melhem
Session5: Lab2: Practice on REST	3	M. Melhem
Session 6: SOAP	2	Aldasht
Session 7: WSDL	**3**	Aldasht
Session8: Lab 3: WSDL practice	3	Aldasht
Session9: ESB	4	Aldasht
Session10: Lab4: Practice on ESB	4	Aldasht
Session11: integration patterns	4	M. Melhem
Session12: Lab5: integration patterns	4	M. Melhem
Session13: BPEL	3	Aldasht
Session14: Lab6: Practice on BPEL	3	Aldasht
Session15: UDDI	2	Aldasht

Session 7: SOAP protocol

Session ILOs

After completing this module students will be able to discuss the:

1. Construction of SOAP messages for web services communication
2. Use SOAP messages for web services communication

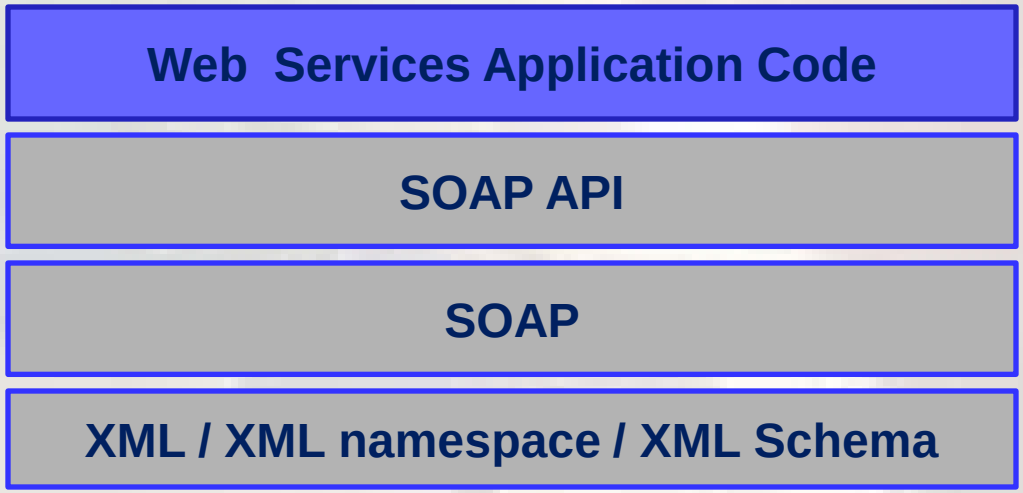
Session Outlines

- **Introduction**
  - **History, definition and basic role**
  - **characteristics**
- SOAP Message format
- SOAP section 5 encoding
- SOAP communication styles
- Summary

Introduction: History

- Microsoft started thinking about XML-based distributed computing in 1997 to enable applications to communicate via RPCs [1].
- In 2000, the XML Protocol working group at the W3C was formed to design the XML protocol “core of XML-based distributed computing”.
- The group started with SOAP 1.1 as a first working draft, then SOAP 1.2 in 2003

SOAP is based on XML concepts



Source, [6]

Introduction: Definition

- Simple Object Access Protocol (SOAP):
  - An XML-based protocol specification for exchanging structured information in the implementation of Web Services [2].
- SOAP relies on other Application Layer protocols, most notably **RPC** and **HTTP** for message negotiation and transmission.
- SOAP can form the foundation layer of a web services protocol stack
- It provides a basic messaging framework upon which web services can be built.

SOAP basic Role

- The role of SOAP [1]:
  - A Web service is a software system designed to support interoperable machine-to-machine interaction.
  - A Web service 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.

SOAP Characteristics

- Provides a mechanism for defining the unit of communication using identifiable SOAP message.
- Provides a processing model:
  - A set of rules for dealing with SOAP messages in software which is the key to use the protocol successfully.
- Provides an extensibility model: using any number of SOAP headers to implement arbitrary extensions on top of SOAP.

SOAP Characteristics, cont.

- Provides a mechanism for error handling
- Provides a flexible mechanism for data representation (text, XML, …)
- Provides a convention for representing Remote Procedure Calls (RPCs) and responses as SOAP messages
- Provides a protocol binding framework: an architecture for building bindings to send and receive SOAP messages over arbitrary underlying transports, HTTP, TCP, UDP, ….

Session Outlines

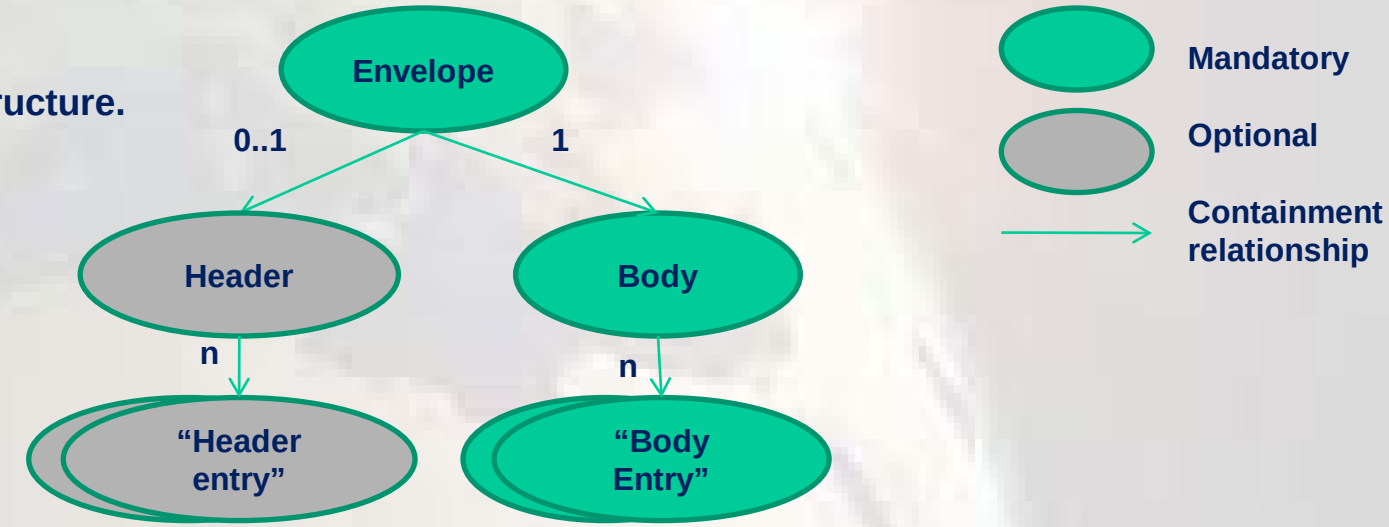
- Introduction
  - History, definition and basic role
  - characteristics
- **SOAP Message format**
- SOAP section 5 encoding
- SOAP communication styles
- Summary

SOAP Message format

- SOAP messages are XML instances consisting of [6]:
  - *Envelope*: a mandatory top element represents the SOAP message and provide a container for header and body.
  - *Header*: an optional element offers a way to pass additional processing or control info.
    - Could convey authentication, QoS and service billing data, and extra *header entries.*
    - If present, it must be the first immediate child.
  - *Body*: a mandatory element carries all mandatory info "*body entries*" for the final recipient

SOAP Message format

SOAP message Containment Structure. Source, [6]



SOAP message Embedded in HTTP Request for `GetMyAgeInDays` Service

```
POST /WebSite2/Service.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/GetMyAgeInDays"
<?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:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
 <GetMyAgeInDays xmlns="http://tempuri.org/">
   <day>int</day>
   <month>int</month>
   <year>int</year>
   </GetMyAgeInDays>
 </soap:Body>
</soap:Envelope>
```

SOAP Message Embedded in HTTP Response for `GetMyAgeInDays` Service

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
<?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:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
   <GetMyAgeInDaysResponse xmlns="http://tempuri.org/">
     <GetMyAgeInDaysResult>int</GetMyAgeInDaysResult>
   </GetMyAgeInDaysResponse>
 </soap:Body>
</soap:Envelope>
```

SOAP message Example

```
<soap-env:Envelope
    xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/Schema"
    xmlns:xsi="http://www.w3.org/2001/Schema-instance">
    <soap-env:Header> <!-- optional -->
        <cXP:priority
        xmlns:cXP="http://companyx.com/ns/priority"
        soap-env:mustUnderstand="1">
        high
        </cXP:priority>
    </soap-env:Header>
    <soap-env:Body>
    <cX:getPhoneNumebr
    xmlns:cX="http://companyx.com/ns/employees"
    soap-env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <first-name xsi:type="xsd:string">Ahmad</first-name>
        <last-name xsi:type="xsd:string">Ahmad</last-name>
    </cX:getPhoneNumber>
    </soap-env:Body>
</soap-env:Envelope>
```

Header Entry

Header Must be processed

Method invocation

Use SOAP encoding

Body Entry

Method Parameters

Method Parameter type

Method Parameter value

Source, [6]

SOAP Message Delivery

- SOAP messages may be handed over from a WS requester to a service provider via *intermediaries*.
- Similar to service providers, intermediaries are identified also via URI value.
- For example, when embedding SOAP into HTTP, the URI maps to the HTTP request URL.

SOAP Message Attributes

- `encodingStyle`: indicates the serialization rules used in the message.
  - Serialization is converting from an application-specific data representation to the wire format.
  - While, deserialization is converting back to the original format.
  - Also, called **marshalling** and **unmarshalling**.
  - Identified through URI in a body entry and applied until another `encodingStyle` attribute appears in the body element e.g. `http://schemas.xmlsoap.org/soap/encoding/`

SOAP Message Attributes, cont.

- `actor`: identifies the application that should process the header entry.

  e.g. `http://schemas.xmlsoap.org/soap/actor/next`
- `mustUnderstand`: identifies whether a SOAP message receiver “target actor” must understand ,value ‘1’, the content of a header entry.

SOAP Message Body Entries

- Recipient must understand and process all body entries.
- Body entries are specific to the application exchanging SOAP messages.
- Recall the phone number request in the example.
- One SOAP-defined body entry exists, is the `Fault` element.
  - Optional child that must not appear more than once within the `body` element.
  - Comprises error and status information.

SOAP Message `Fault` Body Entry Subordinates

- `faultcode`: fault identification
  - `VersionMismatch`: namespace qualification is not identical to: `http://schemas.xmlsoap.org/soap/envelope/`
  - `MustUnderstand`: SOAP application could not process a header entry containing "`MustUnderstand`" with value '1'.
  - `Client`: a SOAP message is not appropriately formed.
  - `Server`: a SOAP message could not be processed
- `faultstring`: human readable fault explanation
- `faultactor`: carries a URI value that identifies the fault originator.
- `detail`: application-specific error info related the body element.

A failing intermediary

- Recall the example of requesting the phone number of Ahmad M. Ahmad.
- Enforced by means of `mustUnderstand`, suppose the intermediary does not understand the header, it responds with fault element:

```
<soap-env:Envelope
   xmlns:soap-
   env="http://schemas.xmlsoap.org/soap/envelope/">
    <soap-env:Body>
     <soap-env:Fault>
      <faultcode>soap-env: MustUnderstand</faultcode>
      <faultstring>
       SOAP MustUnderstand Error </faultstring>
      <faultactor>
        http://companyx.com/messageHub_71</faultactor>
    </soap-env:Fault>
   </soap-env:Body>
</soap-env:Envelope>
```

Visit to see the envelope schema!

http://schemas.xmlsoap.org/soap/envelope/

Session Outlines

- Introduction
  - History, definition and basic role
  - characteristics
- SOAP Message format
- **SOAP section 5 encoding**
- SOAP communication styles
- Summary

SOAP section 5 Encoding

- Connecting heterogeneous applications typically introduces the data-type compatibility problem
- Solution is often based on a common intermediate transfer data format for exchanging information between applications
- SOAP section 5 encoding rules provide a built in mechanism to do that, with appropriate programming languages mappings.
- Using SOAP section 5 encoding is optional, and other encodings may be used as well.

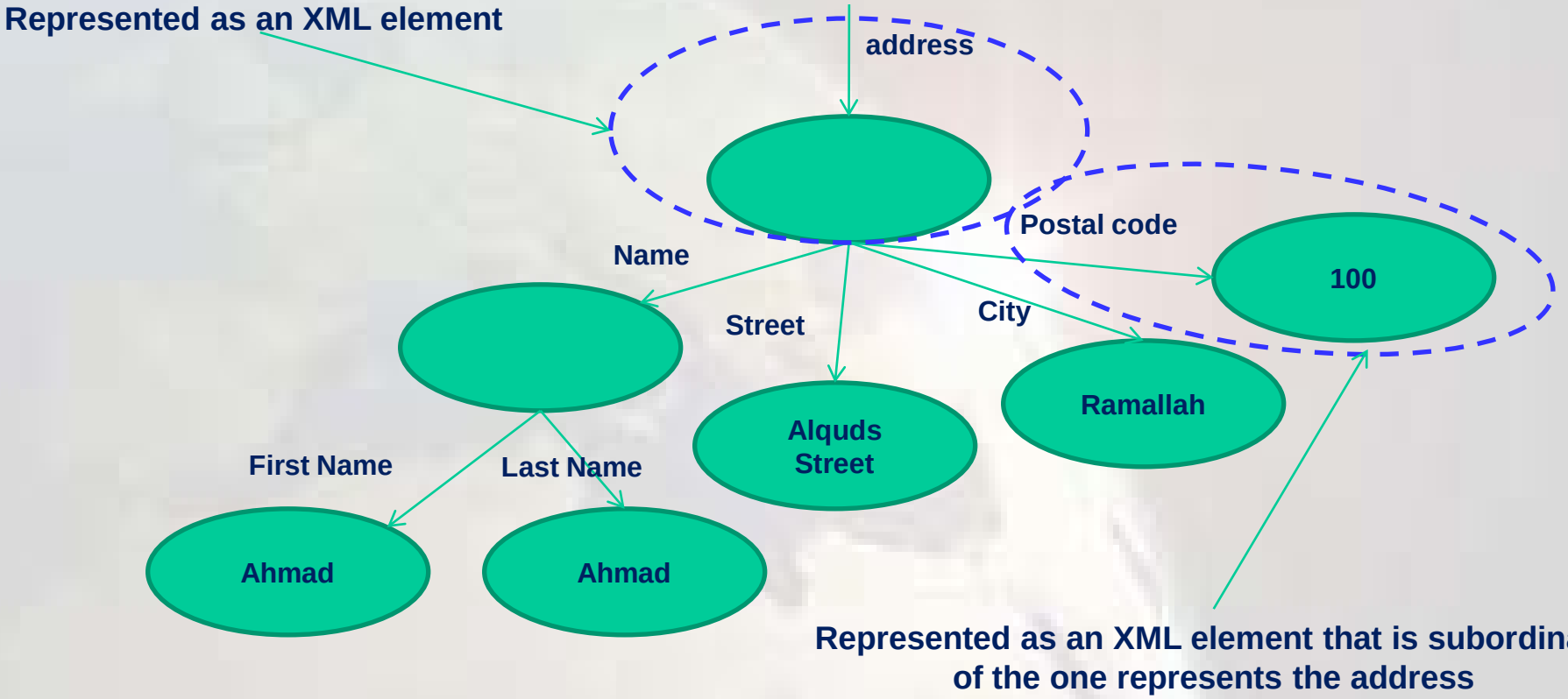
SOAP Values and Data Types

- The SOAP encoding data model consists of simple types and compound types.
- Compound types are based on simple types or other compound types.
- Any application-specific data is represented in terms of a directed graph.

SOAP Values and Data Types, cont.

- Simple value: e.g. string, integer or Boolean, is represented as a node without outgoing edges.
- Compound value: e.g. structure or array, is represented with outgoing edges.
- A simple or compound values may be single-referenced “has only one incoming edge” or multi-referenced “has multiple incoming edge”.
- The SOAP data model is shown in the next slide!

The SOAP Data Model



The SOAP data model, Source, [6] with modification.

SOAP Data Types I

- SOAP encoding adopts all XML schema built-in types.
- But, compound types differ fundamentally from XML schema complex types.
- SOAP types extending the XML schema types are defined in the separate namespace:

http://schemas.xmlsoap.org/soap/encoding/

SOAP Data Types II

- Sender could explicitly assert the type street element content to be a string:
  `<street xsi:type="xsd:string">Alquds Street</street>`
- Or, may not contain the type attribute:
  `<street>Alquds Street</street>`
- SOAP simple values: represented as element content.
  - For each element containing a value, the type is made via XML schema instance `type` attribute.
  - e.g. `<last-name xsi:type="xsd:string">Ahmad</last-name>`

SOAP Data Types III

- SOAP compound values: represented as an element containing a sequence of subordinate elements.
  - SOAP encoding supports two compound types: `structs` and `arrays`
  - e.g. XML encoding for an array in the SOAP object model looks like this:

```
<nameArray xsi:type="SOAP-ENC:array" SOAP-
ENC:arrayType="xsd:string[3]">
    <member>First Name</member>
    <member>Med. Name</member>
    <member>Last Name</member>
</nameArray>
```

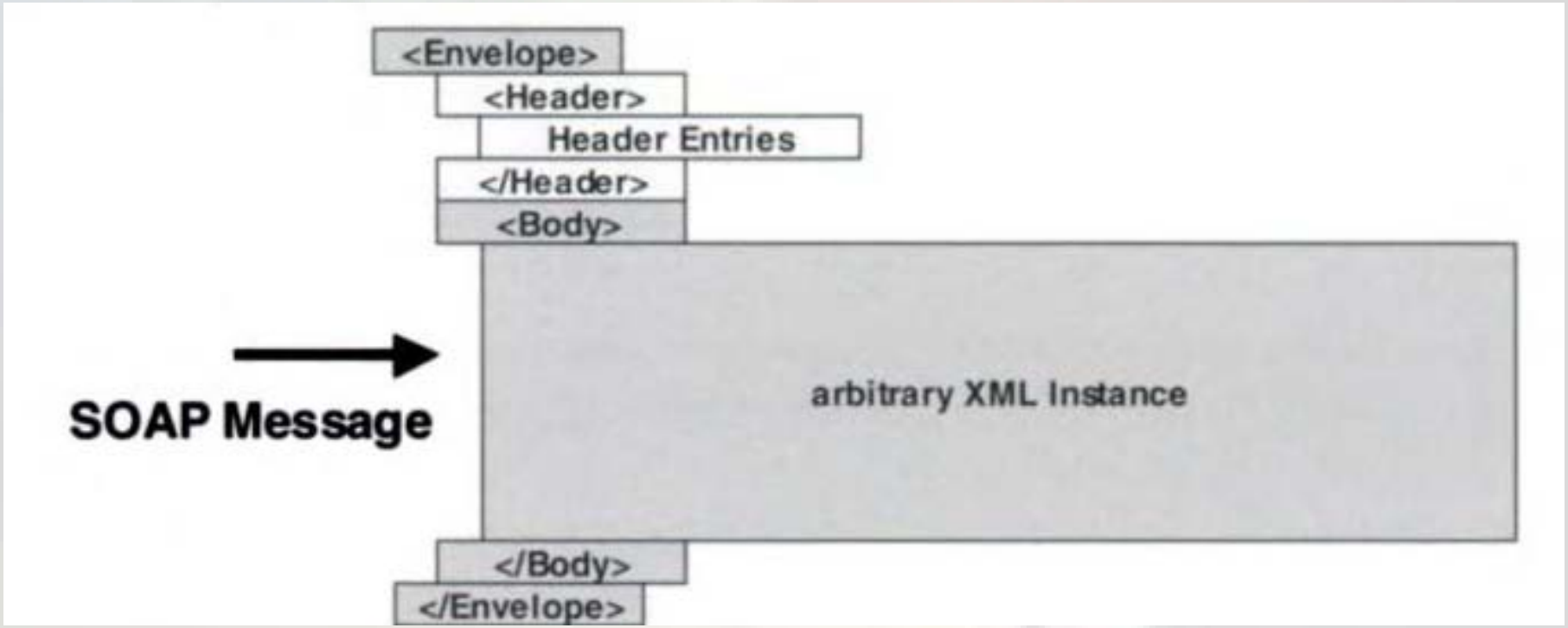
Session Outlines

- Introduction
  - History, definition and basic role
  - characteristics
- SOAP Message format
- SOAP section 5 encoding
- **SOAP communication styles**
- Summary

SOAP Communication style

- SOAP supports two communication styles:
  - Document style: SOAP message body is an arbitrary XML instance.
    - The document style is referred to as message-oriented style.
  - RPC style: represents a remote procedure call.
    - A client invoking a remote procedure expects a result back from the server.
    - Recall our example, it was an RPC style SOAP message
    - The client invoked the method `getPhoneNumber`.

A document style SOAP message

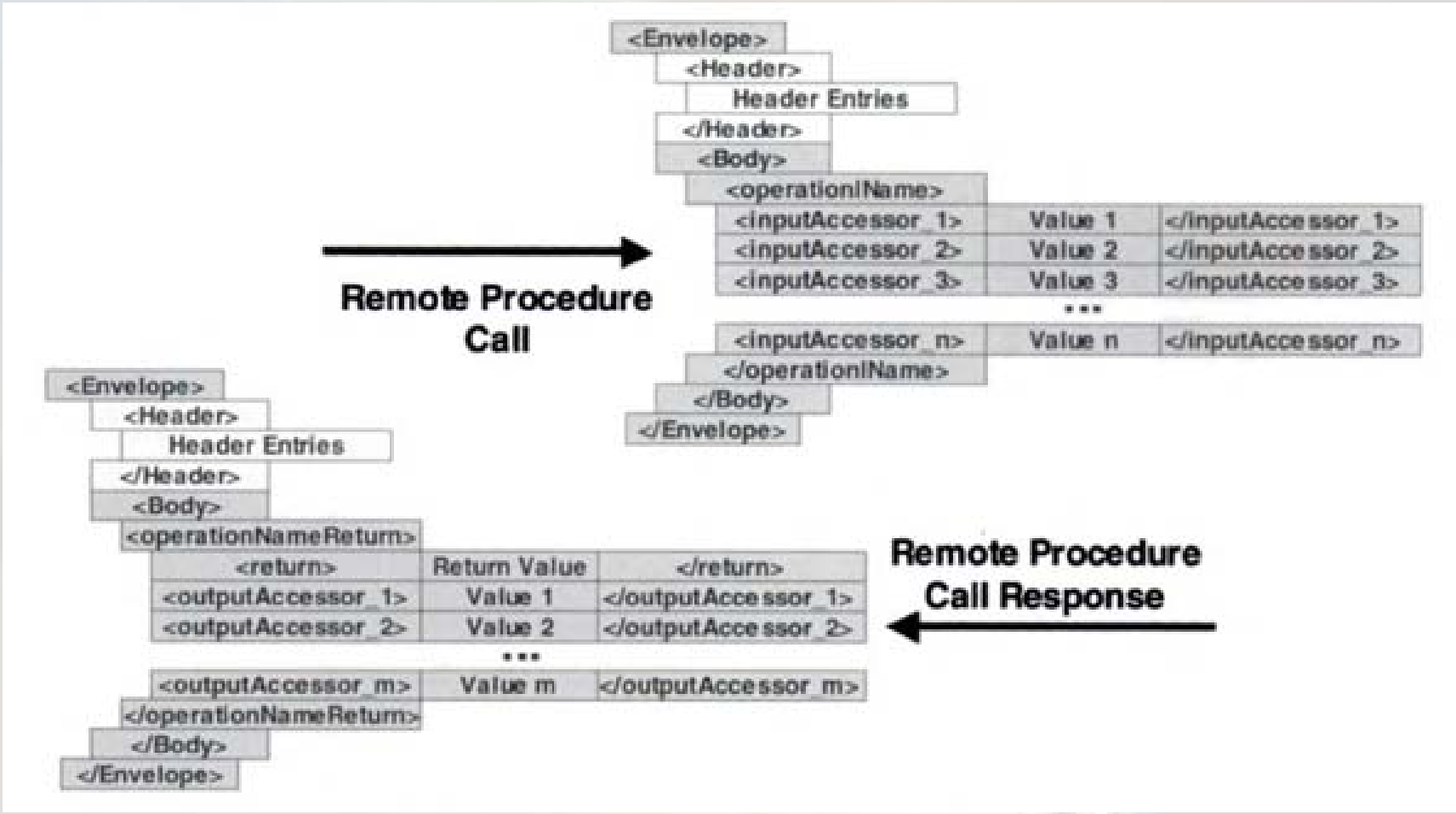


Source, [6]

A document style SOAP message example

```
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <ph:phoneOwner xmlns:ph="http://companyx.com/ns/phoneBook">
      <cX:address xmlns:cX="http://companyx.com/ns/employees"
       targetAddress="PS">
        <cX:name>
          <cX:title selectedTitle="Mr"/>
          <cX:first-name>Ahmad M.<cX:first-name>
          <cX:last-name>Ahmad</cX:last-name>
        </cX:name>
        <cX:street>Alquds Street</cX:street>
        <cX:city>Ramallah<cX:city>
        <cX:postal-code>100</cX:postal-code>
        <cX:country>Palestine</cX:country>
      </cX:address>
      <ph:phonebook>
        <ph:location>Ramallah Office</ph:location>
        <ph:roomNumber>03.04</ph:roomNumber>
        <ph:officePhone>*2900000</ph:officePhone>
      </ph:phone>
    </ph:phoneOwner>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

A SOAP RPC style request and the associated response



Source, [6]

Response on the phone number request from the earlier example (slide 18)

```
<SOAP-ENV:Envelope
 xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/schema"
 xmlns:xsi="http://www.w3.org/2001/schema-instance">
  <SOAP-ENV:Body>
    <companyx:getPhoneNumberReturn
     xmlns:companyx="http://companyx/employees"
     SOAP=ENV:encodingStyle=
     "http://schemas.xmlsoap.org/soap/encoding/">
      <return xsi:type="xsd:string">
        *2900000</return>
    </companyx:getPhoneNumberReturn>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Practices

- Using WCF test client in Visual Studio to test `AgeInDays` service and view the SOAP messages.

Session Outlines

- Introduction
  - History, definition and basic role
  - characteristics
- SOAP Message format
- SOAP section 5 encoding
- SOAP communication styles
- **Summary**

Summary

During this session we have explained the SOAP message construction for web services communication. The following subjects have been covered:

1. SOAP definition and characteristics
2. SOAP Message format
3. SOAP section 5 encoding
4. SOAP communication styles

In the following session we will introduce how to describe a web service using WSDL.

Further reading on SOAP 1.1 can be found at the following link: http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

References

1. Steve Graham, Doug Davis, Simeon Simeonov,Glen Daniels, Peter Brittenham, Yuichi Nakamura, Paul Fremantle,Dieter König and Claudia Zentner, Building Web Services with Java, M A K I N G S E N S E O F X M L , S O A P , W S D L , A N D U D D I, Second Edition, Sams Publishing, 800 East 96th Street, Indianapolis, Indiana 46240, 2005.
2. Extracted from: http://en.wikipedia.org/wiki/SOAP
3. Extracted from http://www.w3.org/TR/soap12-part1
4. Aaron Skonnard, Understanding SOAP, Microsoft Digital Network, “http://msdn.microsoft.com/en-us/library/ms995800.aspx”,March 2003
5. Andrew Lader, Lonnie Wall, Building Web Services and .NET Applications, McGraw-Hill, 2002.
6. Olaf Zimmermann, Mark Tomlinson, Stefan Peuser, “Perspectives on Web services-Applying SOAP, WSDL and UDDI to real-world projects, 2nd edition, Springer, 2005
7. Extracted from: http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

Thanks

Mohammed Aldasht