IT2605
Applications of Web Services

L03 Simple Object Access Protocol (SOAP)

Objectives

- Basics of SOAP
- Advantages and Disadvantages of SOAP
- Architecture of SOAP
- Component of a SOAP Package
- RPC Convention
- SOAP Encoding Rules

Basics of SOAP

- To be able to access an application on a remote computer, we need to have a protocol that is:
  - Simple
  - Light
  - Descriptive
  - Flexible
- SOAP enable applications in a **distributed environment** to be **interoperable** with each other.

http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

Basics of SOAP

- **S**imple **O**bject **A**ccess **P**rotocol
- A **communication** protocol
- For communication **between applications**
- A format for **sending messages**
- Is designed to **communicate via Internet**
- Is **platform independent**
- Is **programming language independent**
- Is based on **XML**
- Is **simple** and **extensible**
- Allows you to **get around firewalls**
- Developed as a **W3C standard**

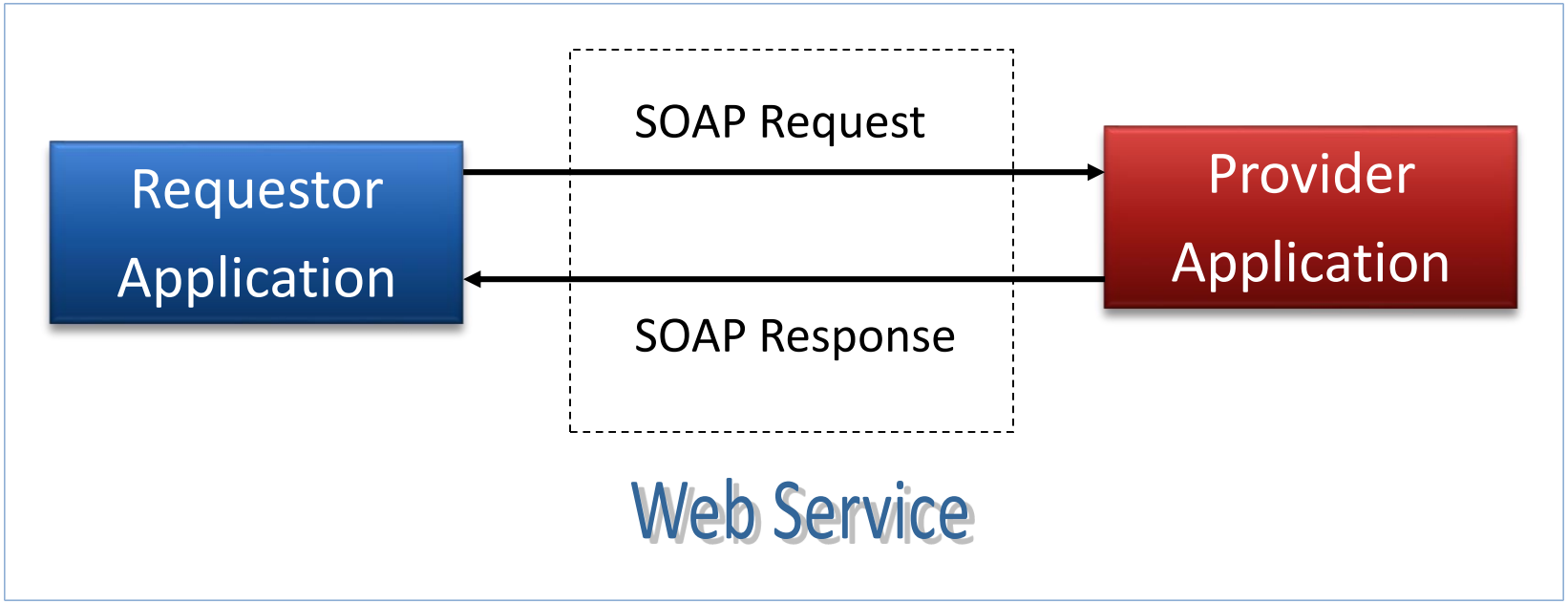
Basics of SOAP

- It is important for application development to allow Internet communication between programs
- Remote Procedure Calls (RPC) between objects like COM, DCOM and CORBA, represents a compatibility and security problem
  - firewalls and proxy servers will normally block this kind of traffic

Basics of SOAP

- A better way to communicate between applications is over HTTP, because:
  - HTTP is widely supported by multiple platforms
  - HTTP traffic is seldom blocked
- SOAP was created to use HTTP
- SOAP provides a way to communicate between applications running on **different operating systems**, with **different technologies and programming languages**
  - Unix, Microsoft Windows Server, iPhone, Android, etc?
  - VB.NET, C#, Java?
  - PC, Mobile Phone, Tablet, etc?

Defining SOAP



The SOAP Protocol

Request - Response Format

Advantages of SOAP

- Supported by major vendors (Microsoft , Oracle and Sun MicroSystems)
- **Widely accepted protocol** for transferring data between distributed applications created using all recent programming languages
- **Uses XML**
  - Transfer data in text format which is self-describing hence enabling interoperability
  - Transfer heavy and complicated data easily by encoding the data
- SOAP messages **often bypass firewalls** (SOAP uses HTTP)
- Data transmitted using **SOAP can be secured** (encryption, HTTP Secure)

Disadvantages of SOAP

- Data need to be ‘packaged’ (SOAP request and message)
- Transferring data using SOAP can be an overhead
  - XML document, DTD & schemas require huge memory and CPU requirement
- Can only test for errors at runtime
- Organisations communicating via Web Services must agree on the standard issues defined in the XML schemas

SOAP Message Building Blocks

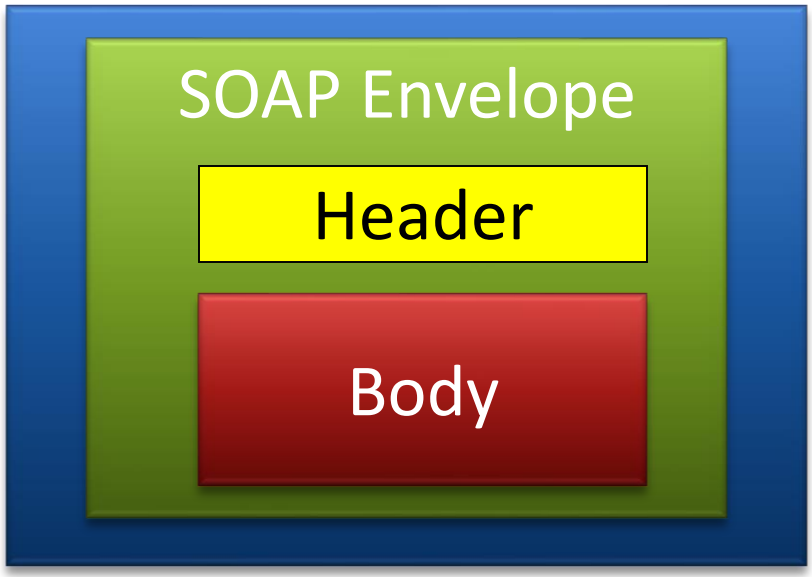
- A SOAP message is an ordinary XML document containing the following elements:
  - A **required** Envelope element that identifies the XML document as a SOAP message
  - An *optional* Header element that contains header information
  - A **required** Body element that contains request and response information
  - An *optional* Fault element that provides information about errors that occurred while processing the message

SOAP Syntax Rules

- A SOAP message **MUST** be encoded using XML
- A SOAP message **MUST** use the SOAP Envelope namespace
- A SOAP message **MUST** use the SOAP Encoding namespace
- A SOAP message must **NOT** contain a DTD reference
- A SOAP message must **NOT** contain XML Processing Instructions

The SOAP Message

- Data is transferred using network protocols (FTP, HTTP)
- Data is stored internally within components of the SOAP message



Skeleton SOAP Message

```
<?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>
        ... ...
    </soap:Header>
    <soap:Body>
        ... ...
        <soap:Fault>
            ... ...
        </soap:Fault>
    </soap:Body>
</soap:Envelope>
```

Envelope

- **Root element** of the SOAP message
- **Required** component
- It **defines the XML document as a SOAP message**.
- Contains **essential information about the messages** that are sent using SOAP
  - Content
  - Information about the sending and receiving application
- Uses the xmlns:soap namespace:
  - http://www.w3.org/2001/12/soap-envelope
  - It defines the Envelope as a SOAP Envelope

Envelope

```
<?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">

            ... Message information goes here ...

</soap:Envelope>
```

If a different namespace is used, the application must generate an error and discard the message.

Envelope

- The **encodingStyle** Attribute
  - used to define the data types used in the document.
  - This attribute may appear on any SOAP element, and it will apply to that element's contents and all child elements.
  - A SOAP message has no default encoding.
  - Syntax :

```
soap:encodingStyle="URI"
```

http://www.w3.org/2001/12/soap-encoding

Header

- **Optional** element
- Provide information in a SOAP message
  - like authentication, payment, etc
  - How receiving application should process the SOAP message
- Can include as many header as required
- If the Header element is present, it must be the first child element of the Envelope element

Note: All immediate child elements of the Header element must be namespace-qualified.

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>
    <m:Trans xmlns:m="http://www.w3schools.com/transaction/"
       soap:mustUnderstand="1">
                    234
    </m:Trans>
  </soap:Header>
    ... ...
</soap:Envelope>
```

Body

- **Required**
- Contains the main text of the message: the actual SOAP message intended for the ultimate endpoint of the message
- All data that needs to be transferred forms a part of the Body
- Denoted by the Body element (child element of Envelope)

Body

- To define text in the Body element – include sub-elements
- Text is encoded using the encodingStyle attribute
- SOAP defines one element inside the Body element in the default namespace (the rest are user/application defined) ("http://www.w3.org/2001/12/soap-envelope")
- SOAP Fault element, which is used to indicate error messages

SOAP Request

```
<?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:GetPrice xmlns:m="http://www.w3schools.com/prices">
        <m:Item>Apples</m:Item>
      </m:GetPrice>
  </soap:Body>
</soap:Envelope>
```

In ASP.NET web services, ASP.NET will create and define these NameSpaces and provide URIs for you.

SOAP 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:GetPriceResponse xmlns:m="http://www.w3schools.com/prices">
               <m:Price>1.90</m:Price>
          </m:GetPriceResponse>
     </soap:Body>
</soap:Envelope>
```

In ASP.NET web services, ASP.NET will create and define these NameSpaces and provide URIs for you.

SOAP Fault Element

- Used to hold error and status information for a SOAP message.
- If a Fault element is present, it must appear as a child element of the Body element
- A Fault element can only appear once in a SOAP message

SOAP Fault sub-Elements

- The SOAP Fault element has the following sub elements:

Sub Element	Description
<faultcode>	A code for identifying the fault
<faultstring>	A human readable explanation of the fault
<faultactor>	Information about who caused the fault to happen
<detail>	Holds application specific error information related to the Body element

SOAP Fault Codes

- The faultcode values defined below must be used in the faultcode element when describing faults:

Error	Description
VersionMismatch	Found an invalid namespace for the SOAP Envelope element
MustUnderstand	An immediate child element of the Header element, with the mustUnderstand attribute set to "1", was not understood
Client	The message was incorrectly formed or contained incorrect information
Server	There was a problem with the server so the message could not proceed

The SOAP Encoding Rules

- Need to adhere to a set of rules when creating a SOAP message
- Defined in SOAP specification called SOAP encoding
  - Similar to XML schemas and define the types and constructs use to create SOAP message
  - Specify the types that you can include in a SOAP message
    - Simple
    - Compound

Types of Data Types

- Simple Data Type
  - Specified in the XML Schema Specification
  - Integer, float, negativeInteger, string, etc
- Compound Data Type
  - Created using one or more simple types
  - Contain child elements
  - 2 types:
    - structs
    - arrays

SUMMARY

- Introduction to SOAP 1.1
- SOAP Building Blocks
- SOAP elements
- Example of SOAP messages

PRACTICAL TIME!