SOAP in 100 minutes

Simple Object Access Protocol

let the world communicate !

---

http://MakSoft.ch

License

Creative Commons

How to use these slides ?

- These Slides were designed to serve as a Tutorial in guided and unguided modes.
- Each Slide attempts to answer the questions raised in the previous slide.
- The slides could also be used as a Desktop Quick Reference.

Please Remember to reference the Author and source of those slides :

Ahmed Maklad

http://MakSoft.ch

Seminar Objectives

- Reminder of XML session.

http://blog.maksoft.ch/2010/10/24/xml-tutorial-in-100-minuits/

- Being SOAP aware. (jargon)
- Able to read and understand WSDL files.
- SOAP VS XML-RPC VS RESTful web-services.
- BPEL : Taking SOAP further for implementing work-flows.
- Learn to use SOAPUI tool, for testing and emulating web-services.
- Learn to use TCPMON tool, for monitoring and troubleshooting TCP/UDP traffic.
- SOAP and AJAX.
- How Does SOAP work in Development
- Sample SOAP development with Perl

Why learn about SOAP ?

- SOAP is a pivotal part in EAI.
- XML and SOAP is a general-use technologies, you might find it anywhere and anytime in a computer-system near you (just like SQL).
- SOAP is a (simple) solution to communicate two programs across networks without being Implementation specific. (Ex. Java→SOAP→.NET)
- SOAP is Human readable!
- SOAP vs. CORBA (ASCII vs Binary)
- SOAP can sneak through Proxies and Firewalls.
- Secured SOAP (HTTP**S** + XML).
- SOAP with attachments. (like HTTP with images inside)
- AJAX wouldn't have been possible without Web Services.
- Web 2.0 is not possible without AJAX.

„Man is the enemy of what he doesn't know". Resolving confusions.

- Every SOAP is a Web-Service.
- But not all Web-Services are SOAP.
- Sometimes a web-service will just offer a response as CSV or Table or even JSON (JavaScript Object Notation).
- Only when the web-service accepts/responds XML according to SOAP standards, we should call it a SOAP-Web-Service.
- SOAP offers inter-operatability between applications.
- SOAP could be used in .NET, Java, Perl ,python JScript ...etc.

What is SOAP ?

- HTTP(S) +XML = SOAP
- **HTTP**rotocol + **XML**anguage = SOAP
- HTTP is an Application Layer protocol (that is layer 7 in OSI network layers model).
-
- The statment: XML+HTTP could also aplly to XML-RPC and REST.
- These slides are focused on SOAP only.
- Other protocols + XML = could also be SOAP.
- For simplicity we should be focused only on HTTP+XML=SOAP.

What is XML ?

- A File-type (TEXT) ?
- A File format ?
- A Language ?
- A Data structure ?
- Maybe Something else?

XML : e**X**tensible **M**ark-up **Language**

So what does exactly XML means?

- A **language** used inside a text formatted file/document to transfer **data** across different Applications.
- This Language has certain **rules** that mold the data you want to transfer into a certain format (a flavour/Vocabulary of XML).
- It is **CaSe SeNsItIvE**, divided into **tags** and **elements**. Each tag marks a beginning and an end of an element. **Attributes** are related to tags.
- Very similar to HTML but with much different purpose and much **stricter rules**.

The design goals for SOAP are:

W3C workgroup.

- http://www.w3.org/TR/2000/NOTE-SOAP-20000508/#_Toc478383487

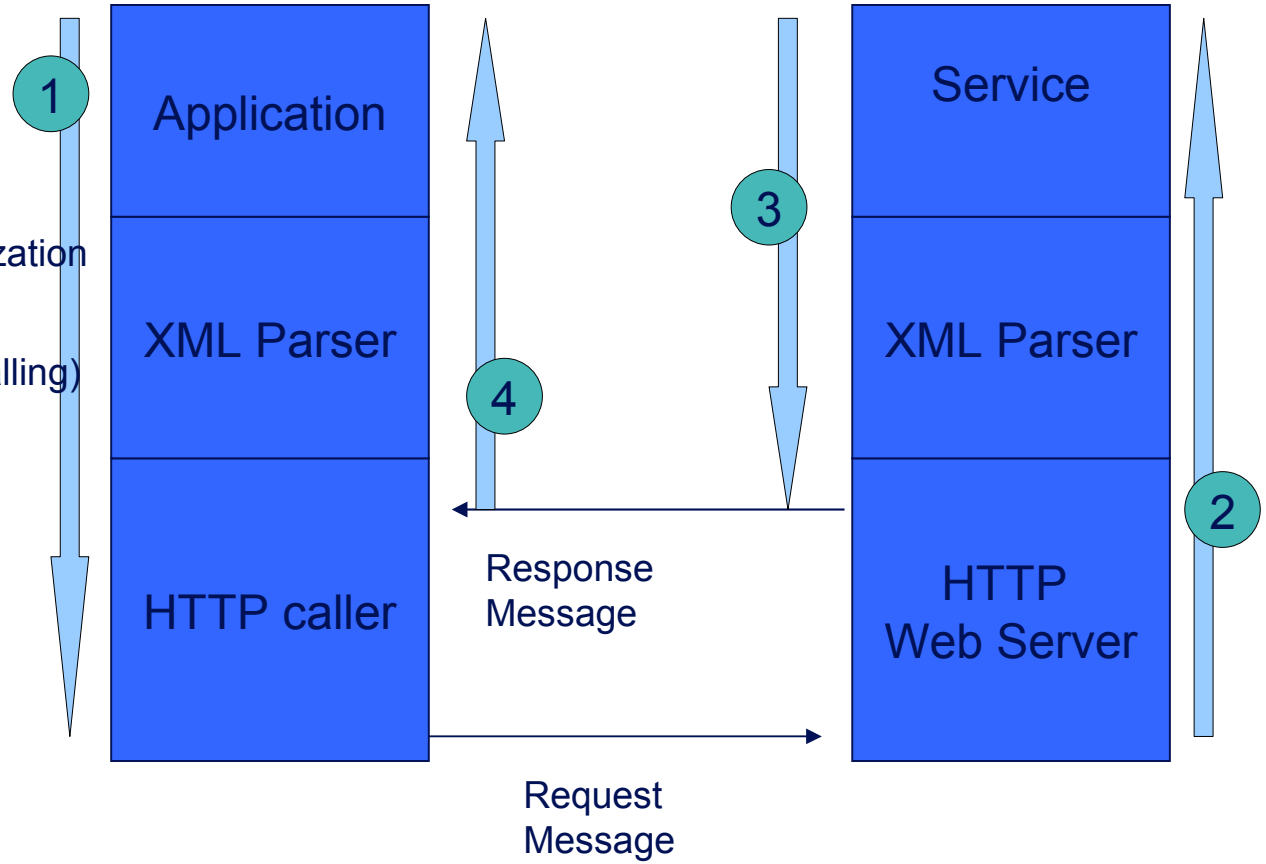
How SOAP works inside applications ?

Memory
Variables /
Objects

Serialization / DeSerialization into XML

(marshalling / demarshalling)

XML wrapped into an HTTP request / response



That's why it is called :

"Simple Object Access Protocol"

What do you need to do SOAP?

	Client WS-Consumer	Server WS-Provider
XML Library to serialize and de-serialize	X	X
HTTP calling library (just like web-browsers)	X	
HTTP web server (a typical HTTP web serer , Apache, TOMCAT,..etc)		X

Inside XML and SOAP syntax

A typical XML document


```
<?xml version="1.0" encoding="UTF-8" ?>

<ROOTELEMENT>
        <SubElement1 attribute11="xyz" attribute12="123">
                CONTENT1
                <SubElement2 attribute21="xyz" attribute22="XYZ">
                        CONTENET2
                </SubElement2>
        </SubElement1>
        <SubElement1 attribute11="xyz" attribute12="123">
                CONTENT3
                <SubElement2 attribute21="xyz" attribute22="XYZ">
                        CONTENET4
                </SubElement2>
        </SubElement1>
        <SubElement1 attribute11="xyz" attribute12="123">
                CONTENT5
                <SubElement2 attribute21="xyz" attribute22="XYZ">
                        CONTENET6
                </SubElement2>
        </SubElement1>
</ROOTELEMENT>
```

XML prolog

One Root Element

One Root Element

SOAP (chain of thoughts)

- **(SOAP) Simple Object Access Protocol**

Communication Protocol

Between Applications

Platform and Language independent

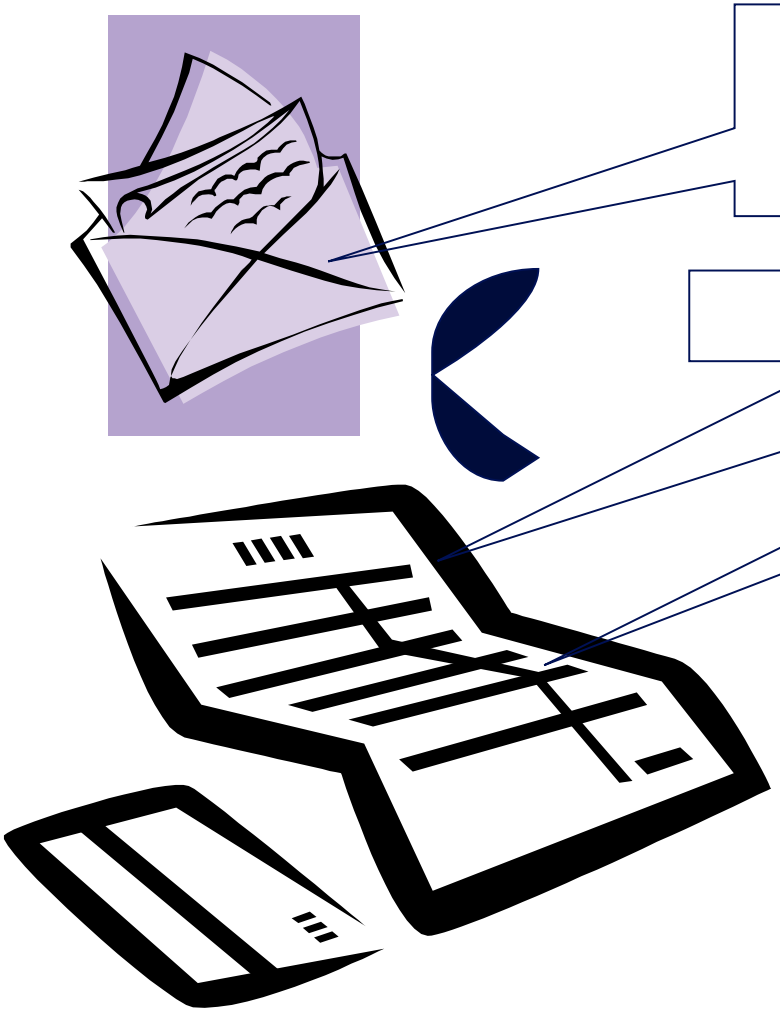
Sending **Message** via Internet or through firewalls

Based on **XML** so it is Simple and easily e**X**tinsible

A W3C Standard

Soap Message

Every Soap Message has a:



Every SOAP message has an envelope which identifies it as a SOAP message.
http://www.w3.org/2001/12/soap-envelope

An **Optional Header**

A **Required** Body

a **Request** Message

a **Response** Message

An **Optional** Fault Message

SOAP rules :

- A SOAP message MUST be encoded using **XML**.
- SOAP message MUST use the SOAP Envelope name-space (**soap**:Envelope).
- A SOAP message MUST use the SOAP Encoding name-space (**soap**:encodingStyle).
- A SOAP message must NOT contain a DTD reference (maybe XSD)
- A SOAP message must NOT contain XML Processing Instructions

Name Spaces
(part of XML Schema specs)

- Sometimes you need to use two different elements with the same name but they represent different structures. And we need somehow to include both elements in one XML document.

Example :

- Customer element for Billing system.
- Customer element for CRM system.

Each has different data associated to it different from the other.

- One way to overcome this problem is to use the same element name, but prefix it with something more distinctive.

Example:

- BILLSYS:CUSTOMER
- CRMSYS:CUSTOMER

- But we have to explicitly mention every time we use this confused-about-its-origin element to use which definition of that element:

```
<BILLSYS:CUSTOMER xmlns:BILLSYS="c:\..\..">
    <BILLSYS:name>MyNameIsSoAndSo</BILLSYS:name>
    <BILLSYS:address>MyStreet StreetNumber</f:address>
</BILLSYS:table>
```

NameSpaces (cont.)

- “xmlns:=..” is an attribute of the start tag of an element.

```
xmlns:namespace-prefix="namespaceURI"
```

- When a namespace is defined in the start tag of an element, all child elements with the same prefix are associated with the same namespace
- Note that the address used to identify the namespace is not used by the parser to look up information. The only purpose is to give the namespace a unique name.

A sample 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>
```

Mandatory Element which is the **root element** of the SOAP message.

Note the extensive use of Name-Spaces

It has to be this Name-Space otherwise the message will be rejected!

Optional, Must be the first element after the Envelope and contains Application Specific data, Authentication, payment ..etc

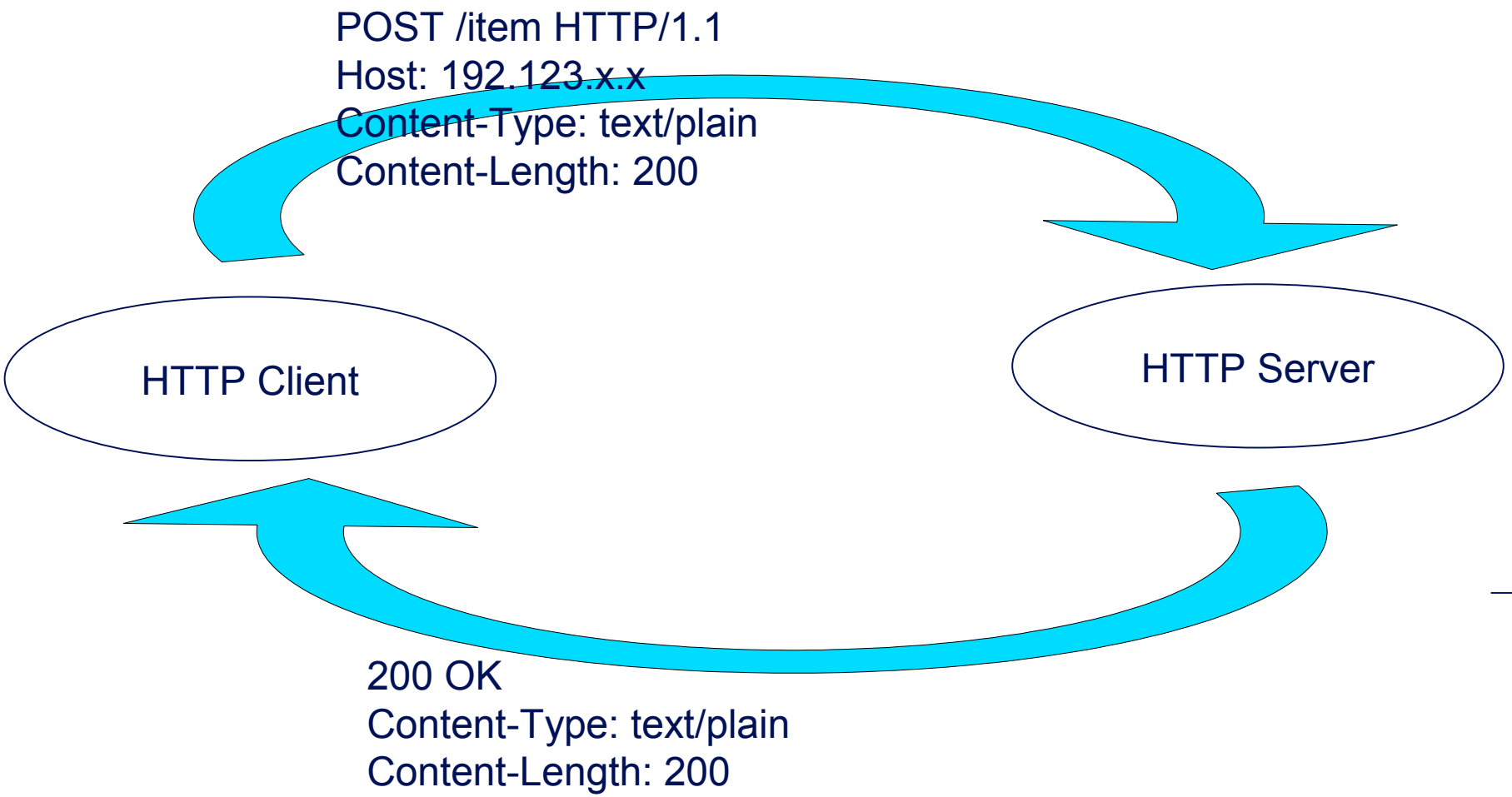
SOAP Messaging Request and Response


```
<?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:Header> … </soap:Header>
   <soap:Body>
      <m:GetPrice
        xmlns:m="http://www.w3schools.com/prices">
        <m:Item>Apples</m:Item>
      </m:GetPrice>
   </soap:Body>
</soap: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">
<soap:Body>
<m:GetPriceResponse xmlns:m="http://www.w3schools.com/prices">
  <m:Price>1.90</m:Price>
</m:GetPriceResponse>
 </soap:Body>
</soap:Envelope>
```

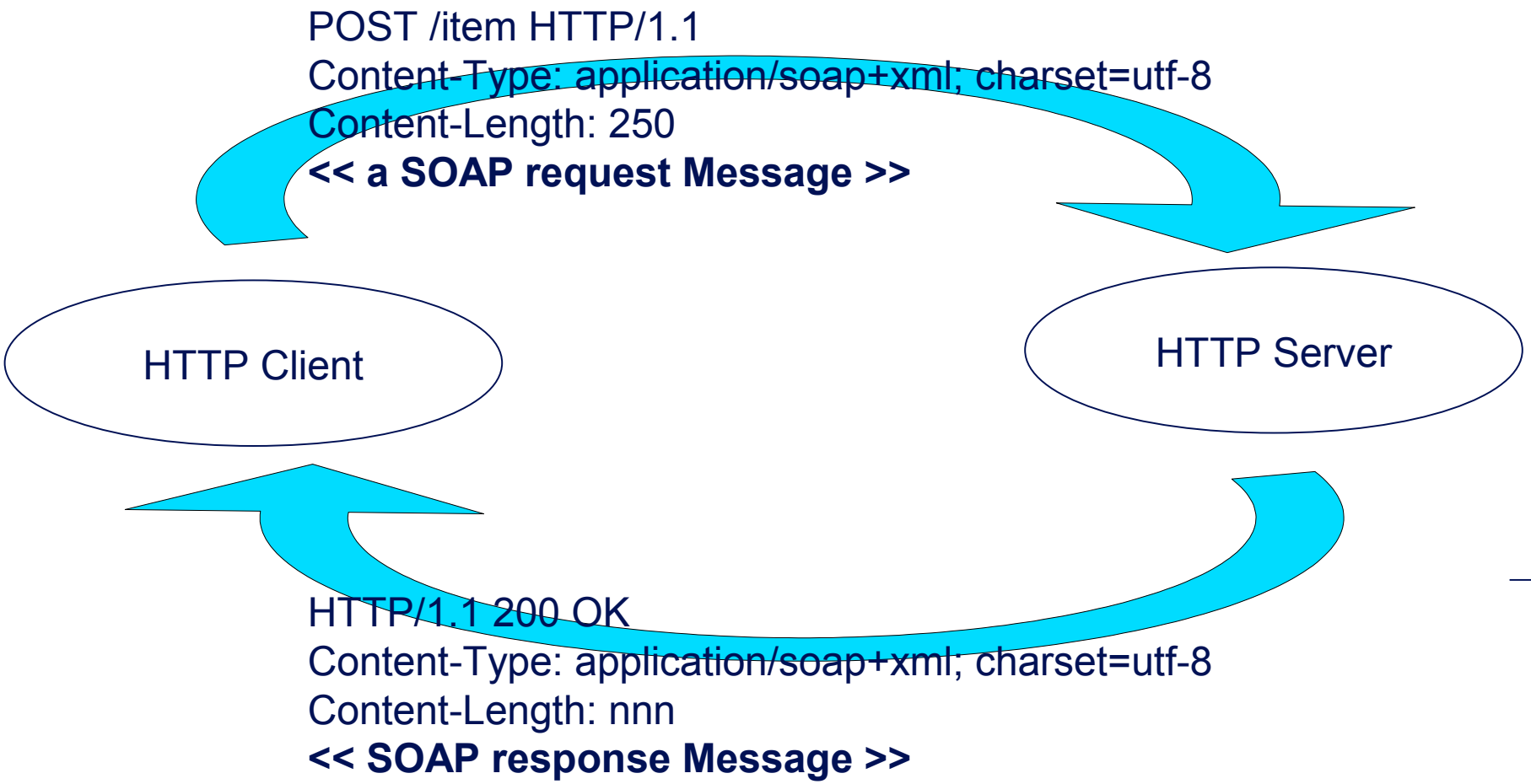
SOAP HTTP binding: (HTTP + SOAP=Web Services)

- **A normal HTTP request:**



SOAP HTTP binding: (HTTP + SOAP=Web Services)

- **A normal HTTP/SOAP request:**



Body\Fault Section
Sometimes things go wrong!

- By default if something goes wrong on the Server side (Exceptions)
- The server send back a fault response not the normal response message.
- The SubElements of the SOAP fault message are as follows:

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 The detail element is intended for carrying application specific error information related to the Body element.

Sample fault response


```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
  <SOAP-ENV:Fault>
            <faultcode>SOAP-ENV:Server</faultcode>
            <faultstring>This is an operation implementation generated fault</faultstring>
            <faultactor />
            <detail>
                        <ns:searchCustomerFault
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ns="http://www.thisschema.com/ns/ONECRM/BSCS/customer/search/v1-0"
  xmlns:ns0="http://schemas.xmlsoap.org/soap/envelope/">
                        <ns:ErrorCode>-99</ns:ErrorCode>
                        <ns:ErrorDescription>A timeout has occurred</ns:ErrorDescription>
                        </ns:searchCustomerFault>
            </detail>
  </SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Web Service Descriptor Language (WSDL)

Web Service Descriptor Language (WSDL)

- WSDL is a document written in XML. The document describes a Web service. It specifies the **location** of the service and the **operations** (or methods) the service exposes.
- It is like the *.h (header) file in C/C++, where you define the interfaces (descriptor of in/output) of various functions/objects used for the actual library or DLL (contains logic) you will use.

WSDL definition

- *W3C Abstract:*
- "WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information.
- The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint.
- Related concrete endpoints are combined into abstract endpoints (services).
- WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate, however, the only bindings described in this document describe how to use WSDL in conjunction with SOAP 1.1, HTTP GET/POST, and MIME."

Defining Services in WSDL

Services are defined using six major elements:

- **types**, which provides data type definitions used to describe the messages exchanged.
- **message**, which represents an abstract definition of the data being transmitted. A message consists of logical parts, each of which is associated with a definition within some type system.
- **portType**, which is a set of abstract operations. Each operation refers to an input message and output messages.
- **binding**, which specifies concrete protocol and data format specifications for the operations and messages defined by a particular portType.
- **port**, which specifies an address for a binding, thus defining a single communication endpoint.
- **service**, which is used to aggregate a set of related ports.

Analogy (Function <=> PortTypes)

- If a **Service** is a Library(DLL/JAR...etc) and the functions it offers are **PortTypes**.
- Then the list of arguments for each Function is a **Messages**.
- And the actual arguments field-types are **Types**.
- **Fault** would be the kind of exceptions this function troughs in case of problems.
- **Binding** is the glue that binds these entities into meaningful context. Defining exactly which request-Message is sent to which PortType and which response-Message is expected as a result and which fault could be dispatched from which PortType.
- **Endpoints** define where to find the Service. (mostly PROTOCOL://HOST:PORT)
- A WSDL file defines types,Messages,portTypes and faults individually and then define the binding and the service.

XML Elements of WSDL file

Element	Description
wsdl:types	Container element for data type definitions that are made using XML Schema (XSD) or another similar system for data types. The data types used by the web service
wsdl:message	Definition of the message data being communicated. The message can be made up of multiple parts and each part can be of a different type, The messages used by the web service
wsdl:portType	Abstract set of operations supported by one or more endpoints. The operations performed by the web service
wsdl:binding	Concrete protocol and data format specification for a particular port type. The communication protocols used by the web service
wsdl:service	Collection of related endpoints.

WSDL XML elements

Root Element of WSDL files

```
<definitions>

   <types> definition of types........ </types>

   <message> definition of a message.... </message>

   <portType> definition of a port....... </portType>

   <binding> definition of a binding.... </binding>

   <!-- also possible optional element →

  <service>  defininition of Services .. </service>

</definitions>
```

data elements of an operation. Each message can consist of one or more parts. The parts can be compared to the parameters of a function call in a traditional programming language.

the most important WSDL element. element can be compared to a function library (or a module, or a class) in a traditional programming language.

message format and protocol details for each port.

WSDL ERD

Source : http://www.w3.org/TR/2007/PR-wsdl20-primer-20070523/

Example WSDL

Let us play match the colors game.:


```
<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>
```

On the Server Side

getTerm (String **term**) return String **Value**

Operation Types

- A WSDL port describes the interfaces (legal operations) exposed by a web service.
- There are various types of operations/Functions

Type	Definition
One-way	The operation can receive a message but will not return a response (Procedure) Only an <input> element in the <operation> section.
Request-response	The operation can receive a request and will return a response (Function) <input> and <output> element in the <operation> section.
Solicit-response	The operation can send a request and will wait for a response (Synchronous Event?)
Notification	The operation can send a message but will not wait for a response. (AsSynchronous Event?)

Binding

- WSDL bindings defines the message format and protocol details for a web service.
- Is the glue that binds everything together and provides meaning to all of the definistions.

Binding Example


```
<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>

<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>
```

Points to the port of the binding

You can name it anything you want

rpc / document

The SOAP protocole : HTTP,HTTPS,SMTP,..etc

The action to be called to perform this operation

How the input and output are encoded : literal,BASE64,..etc

XMP-RPC vs SOAP

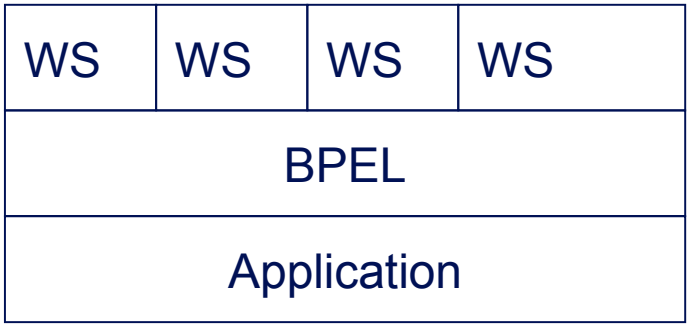
- XML-RPC : is very humble in its goals, only to provide a way to remotely calling procedures.
- SOAP : is a way to exchange Objects. And Messages.
- XML-RPC : does not require WSDL files.
- The main difference is the data-types used in each protocol.
- Both are HTTP+XML , but SOAP allows more complex data-types.
- SOAP is newer than XML-RPC.

SOAP vs REST

- REST is newer than SOAP.
- REST simplifies the request and response: http://www.thersswebblog.com/?guid=20060704042846
- REST is more bound to HTTP, while SOAP could use other protocols than HTTP.
- If you are designing a new web-service then is preferable to us REST rather than SOAP.
-

BPEL (Business Process Execution Language)

- Purpose
- Structure
- EAI
- Is a web-service
- BPEL simplifies the work-flow of WS (Orchestrates).
- Work-flow management
- Finally described in an XML file (something like a WSDL)



BPEL (Business Process Execution Language)

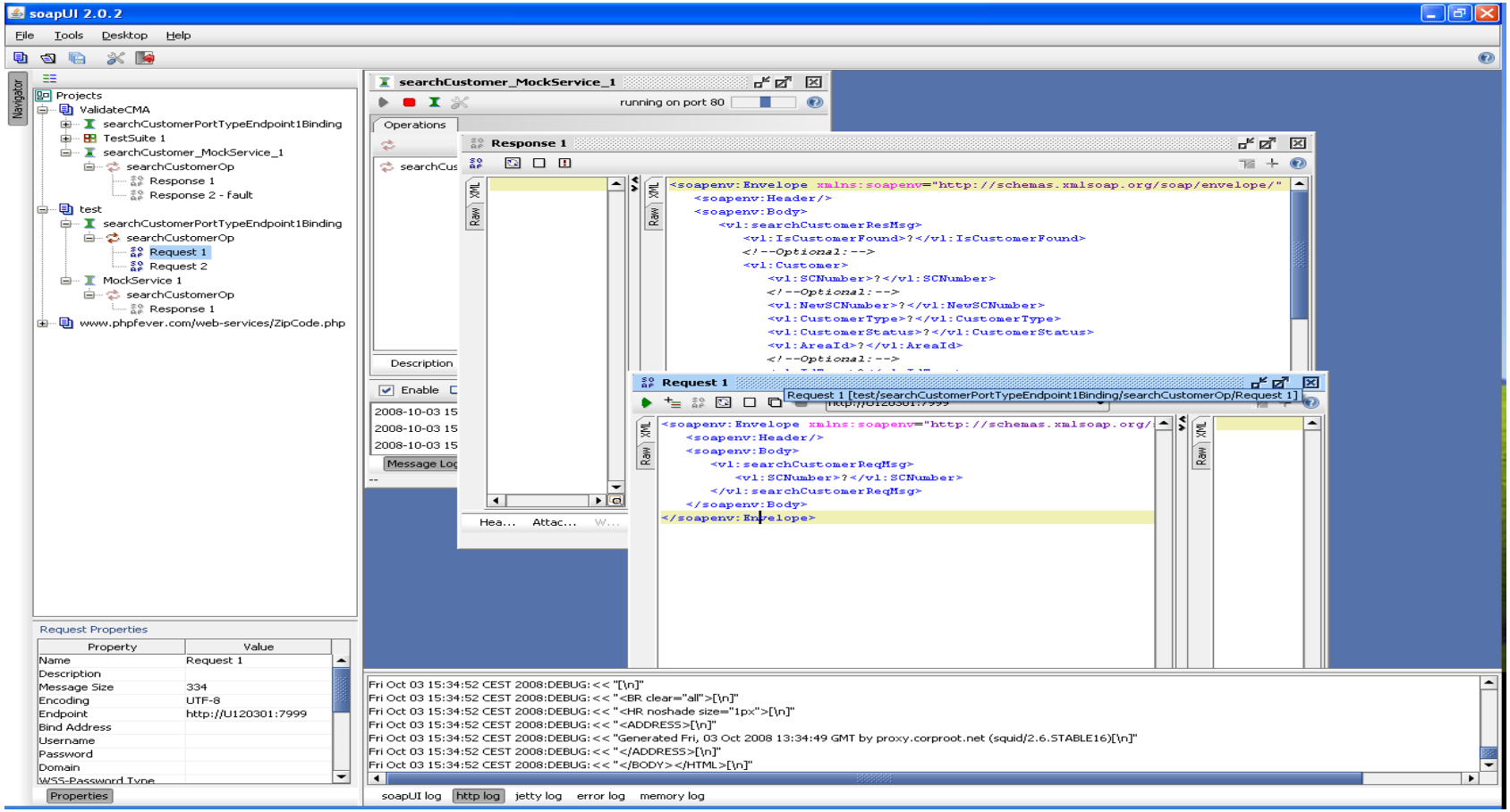
- WS provides individual business logic in a very nice way.
- Like functions each porttype inside a WS is rather atomic.
- BPEL offers a way to orchestrate how different Functions work together towards a unified objective (Workflow).
- Each function could be used in more than one work-flow.
- An Example:
- You have multiple functions available like:
  - Check Credit
  - Check Stock
  - Reserve Stock.
- A customer wanting to buy from the stock would have to go through all of these functions to be able to finally get a confirmation.
- BPEL defines the Path or flow of procedures and conditions for call the various WS in a certain work-flow

Useful SAOP Tools

- SOAPUI
- TCPMON

SOAPUI.org tool
http://www.soapui.org/

- A tool that can generate SOAP request and record responses.
- Can act as a service and simulate a response for requests.



Using SOAPUI to test a WS

- File → new WSDL project → load the WSDL file.
- Navigate to the operation you want to test and right-click, « newrequest «,You get a sample Request Message.
- In the upper selectBox choose « Edit Current » and type in the address of the web server you would like to test.
- Submit the request through the Green arrow button on the top left of the Requets window and examin the response message.
- Manipulate the request message as you desire, and values, and then resend the modified request.
- Note: if you access Internet WS, you need to setup the proxy and « Excludes » list properly.

Using SOAPUI to simulate a WS mocking

- When you like one of the resposnes and you would like to have your own server that simulates it, you can right-click it or the operation and choose « add to mockservice »,You get a new Node with ResponseX.
- Right click the mock service and « Show MockService Editor».
- Adjust the port which you want to Service request on an dthen click the Green arrow button to start the web service.
- Now you can test you developed application with SOAPUI as a simulator.

TCPMON tool

- https://tcpmon.dev.java.net/
- Sometimes you need to spy on the TCP package for troubleshooting or learning purposes.
- TCPMON acts (more or less) as a TCP proxy.
- You start TCPMON jar file and configure it to accept TCP traffic on a certain port. And you also tell it that all traffic accepted on that port should be forwarded to a certain destination (address/url/ip:port).
- It will just record the TCP session requests ,forward as it is to the destination and take the response and pass it to the original requester.
- After the TCP session is finished , it will display the request and response.

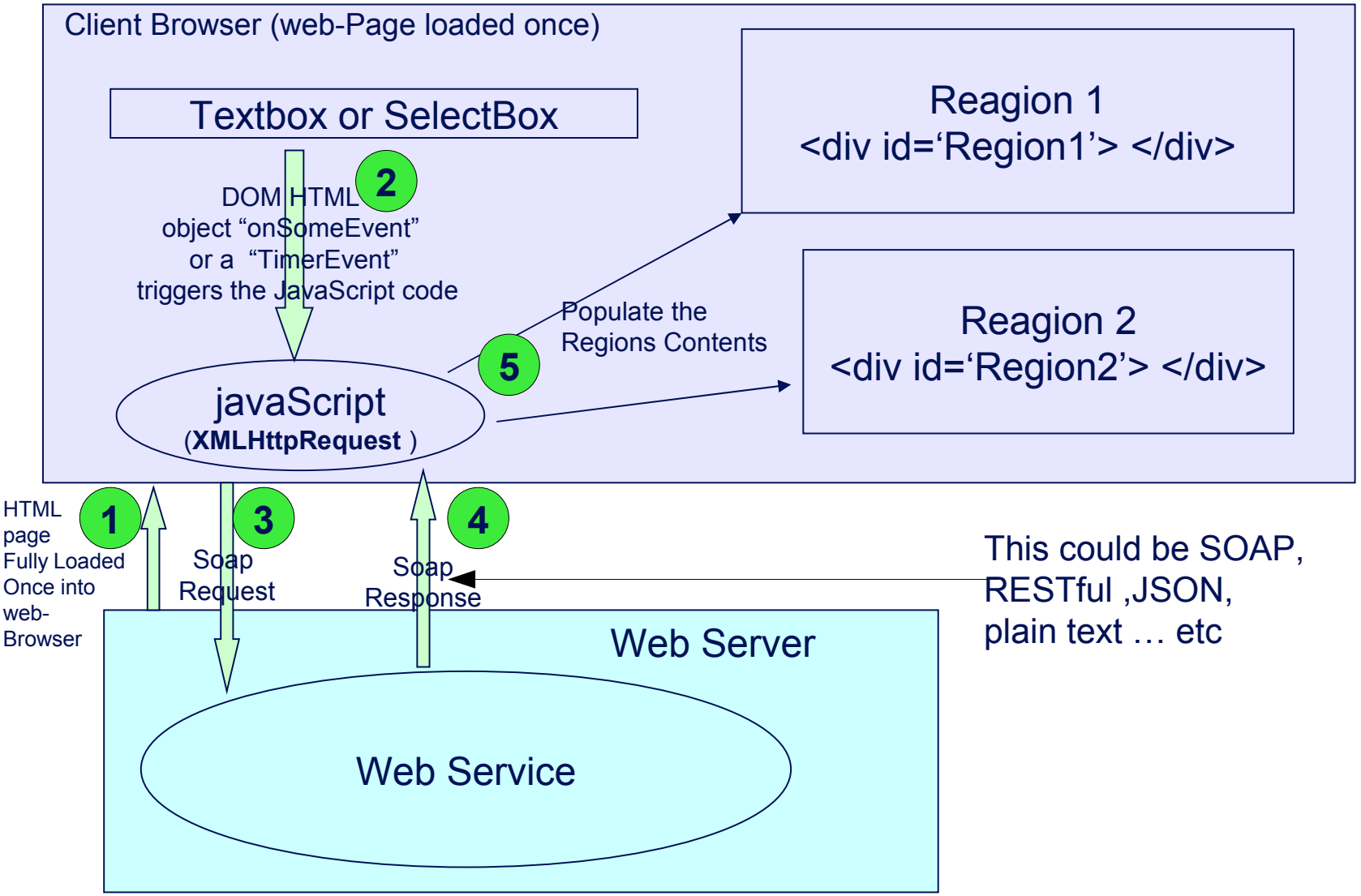
Practical implementations of SOAP

SOAP applications in Real Life

- RSS feeds
- AJAX (GMAIL, Web 2.0 ..etc)
- RPC for applications from different platforms.
- A typical EAI-BUS/EBS/SOA depends on SOAP/WS or CORBA protocols to communicate.

AJAX

Asynchronous JavaScript And XML



How does SOAP works in programming.

- Usually a Class is created and this class becomes the Service.
- Some of the public methods or functions in the service becomes exposed to the outside world as portTypes.
- Some Auxiliary Classes are created to define the Object **Types** to be used for communication (remember S**O**AP ?).
- Optional:Now one way or another from the code the Classes and their methods are tagged as web-services/portTypes and execution environment starts accepting requests (Example:Java @annotation).
- There are two ways to start developing a SOAP web-service:
  - Start by building server-side-classes then create the WSDL file then Client.
  - Start by Building or acquiring a WSDL and then have some tool to create the Classes stubs then fill in the functionality in the server side classes then create the the client WS-consumer.
- Since this is a **Web**-Service, a **Web**-server is needed to run the server-side code.

Perl Sample SOAP Server


```
#!perl –w
# Sample Soap server
# file to be saved under : <xampp>/htdocs/soap_perl/hibye.cgi
 use SOAP::Transport::HTTP;
 SOAP::Transport::HTTP::CGI -> dispatch_to('Demo') -> handle;
# Class Demo is the service and provides two operations
 package Demo;
 sub hi {
   return "hello, world";
}
 sub bye {
   return "goodbye, cruel world";
}
```

Perl Sample SOAP Client


```
#!perl -w
use SOAP::Lite;
print SOAP::Lite -> uri('http://www.soaplite.com/Demo') ->
  proxy('http://localhost/soap_perl/hibye.cgi') -> hi() -> result;
```

SOAP::Lite Object

- uri() identifies the class to the server, and the proxy() identifies the location of the server itself.
- **proxy()**
  - is simply the address of the server to contact that provides the methods. You can use http:, mailto:, even ftp: URLs here.
- **uri()**
  - Each server can offer many different services through the one proxy() URL. Each service has a unique URI-like identifier, which you specify to SOAP::Lite through the uri() method. If you get caught up in the gripping saga of the SOAP documentation, the "namespace" corresponds to the uri() method.

Where to learn more about XML & SOAP ?

- http://www.w3schools.com/default.asp
- http://www.w3.org
- http://www.xml.com/
- http://www.xmlfiles.com/xml/
- http://java.sun.com/xml/tutorial_intro.html
- http://www.brics.dk/ixwt/
- Just google it …

For more Tutorials, Go to:

http://blog.maksoft.ch/tutorials

YOUR OPINION MATTERS !

Comments and feedback about this tutorials are appreciated.

Disclaimer

- All data and information provided on this site are for informational purposes only. Ahmed Makald, Maksoft.ch and sub-domains makes no representations as to accuracy, completeness, currentness, suitability, or validity of any information in this document and will not be liable for any errors, omissions or delays in this information or any losses, injuries, or damages arising from its display or use. All information is provided on an as-is basis.