Web services

Internet Technology CSC457

Web Services Tutorial

WSDL

WSDL stands for Web Services Description Language

WSDL is an XML-based language for describing Web services.

WSDL is a W3C recommendation

SOAP

SOAP stands for Simple Object Access Protocol

SOAP is an XML based protocol for accessing Web Services.

SOAP is based on XML

SOAP is a W3C recommendation

UDDI

UDDI stands for Universal Description, Discovery and Integration

UDDI is a directory service where companies can search for Web services.

UDDI is described in WSDL

UDDI communicates via SOAP

RDF

RDF stands for Resource Description Framework

RDF is a framework for describing resources on the web

RDF is written in XML

RDF is a W3C Recommendation

http://www.w3schools.com/webservices/default.asp

Introduction to Web Services

Web Services

- Web services are application components
- Web services communicate using open protocols
- Web services are self-contained and self-describing
- Web services can be discovered using UDDI
- Web services can be used by other applications
- HTTP and XML is the basis for Web services

Interoperability has Highest Priority

- When all major platforms could access the Web using Web browsers, different platforms couldn't interact. For these platforms to work together, Web-applications were developed.
- Web-applications are simply applications that run on the web. These are built around the Web browser standards and can be used by any browser on any platform.

Web Services take Web-applications to the Next Level

- By using Web services, your application can publish its function or message to the rest of the world.
- Web services use XML to code and to decode data, and SOAP to transport it (using open protocols).
- With Web services, your accounting department's Win 2k server's billing system can connect with your IT supplier's UNIX server.

Web Services have Two Types of Uses

- **Reusable application-components.**
  - There are things applications need very often. So why make these over and over again?
  - Web services can offer application-components like: currency conversion, weather reports, or even language translation as services.
- **Connect existing software.**
  - Web services can help to solve the interoperability problem by giving different applications a way to link their data.
  - With Web services you can exchange data between different applications and different platforms.

Web Services Example

Any application can have a Web Service component.

Web Services can be created regardless of programming language.

A Web Service Example

In the following example we will use ASP.NET to create a simple Web Service that converts the temperature from Fahrenheit to Celsius, and vice versa:

```
<%@ WebService Language="VBScript" Class="TempConvert" %>

Imports System
Imports System.Web.Services

Public Class TempConvert :Inherits WebService

< WebMethod()> Public Function FahrenheitToCelsius
(ByVal Fahrenheit As String) As String
  dim fahr
  fahr=trim(replace(Fahrenheit,",","."))
  if fahr="" or IsNumeric(fahr)=false then return "Error"
  return ((((fahr) - 32) / 9) * 5)
end function

< WebMethod()> Public Function CelsiusToFahrenheit
(ByVal Celsius As String) As String
  dim cel
  cel=trim(replace(Celsius,",","."))
  if cel="" or IsNumeric(cel)=false then return "Error"
  return ((((cel) * 9) / 5) + 32)
end function
end class
```

This document is saved as an .asmx file. This is the ASP.NET file extension for XML Web Services.

Web Services Example

Example Explained

Note: To run this example, you will need a .NET server.

The first line in the example states that this is a Web Service, written in VBScript, and has the class name "TempConvert":

<%@ WebService Language="VBScript" Class="TempConvert" %>

The next lines import the namespace "System.Web.Services" from the .NET framework:

Imports System
Imports System.Web.Services

The next line defines that the "TempConvert" class is a WebService class type:

Public Class TempConvert :Inherits WebService

The next steps are basic VB programming. This application has two functions. One to convert from Fahrenheit to Celsius, and one to convert from Celsius to Fahrenheit.

The only difference from a normal application is that this function is defined as a "WebMethod()".

Use "WebMethod()" to convert the functions in your application into web services:

Web Services Example

```
<WebMethod()> Public Function FahrenheitToCelsius
(ByVal Fahrenheit As String) As String
  dim fahr
  fahr=trim(replace(Fahrenheit,",","."))
  if fahr="" or IsNumeric(fahr)=false then return "Error"
  return ((((fahr) - 32) / 9) * 5)
end function

< WebMethod()> Public Function CelsiusToFahrenheit
(ByVal Celsius As String) As String
  dim cel
  cel=trim(replace(Celsius,",","."))
  if cel="" or IsNumeric(cel)=false then return "Error"
  return ((((cel) * 9) / 5) + 32)
end function
```

Then, end the class:

```
end class
```

Publish the .asmx file on a server with .NET support, and you will have your first working Web Service.

Look at our example Web Service

ASP.NET Automates the Process

With ASP.NET, you do not have to write your own WSDL and SOAP documents.

If you look closer at our example Web Service, you will see that ASP.NET has automatically created a WSDL and SOAP request.

Web Services Example

Using the Web Service Example

In the previous page we created a Web service.

The FahrenheitToCelsius() function can be tested here: FahrenheitToCelsius

The CelsiusToFahrenheit() function can be tested here: CelsiusToFahrenheit

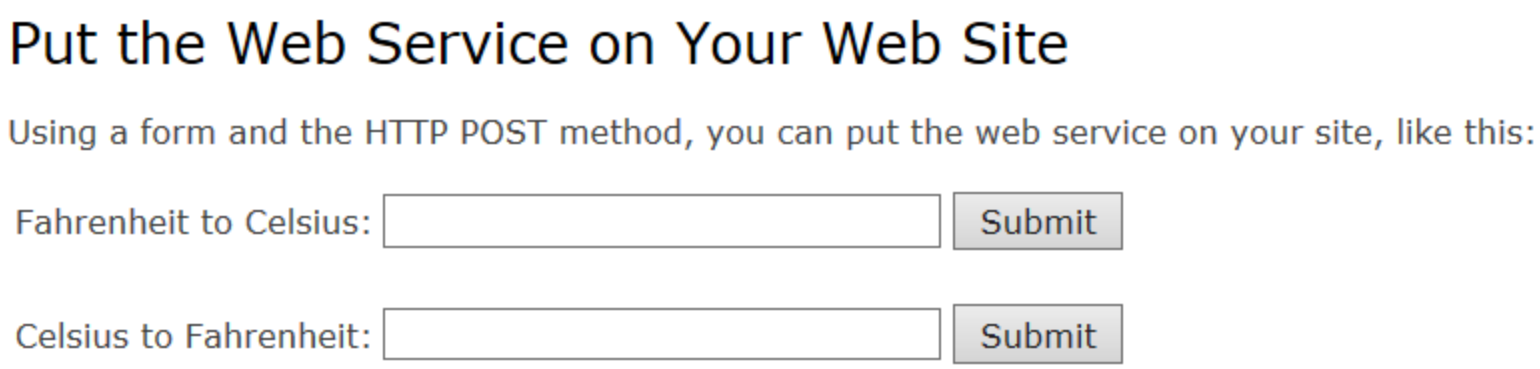
These functions will send an XML response like this:

<?xml version="1.0" encoding="utf-8" ?>
< string xmlns="http://tempuri.org/">38</string>

Web Services Example

Put the Web Service on Your Web Site

Using a form and the HTTP POST method, you can put the web service on your site, like this:



Web Services Example

How To Do It

Here is the code to add the Web Service to a web page:

```
<form action='tempconvert.asmx/FahrenheitToCelsius'
method="post" target="_blank">
< table>
  <tr>
    <td>Fahrenheit to Celsius:</td>
    <td>
    <input class="frmInput" type="text" size="30" name="Fahrenheit">
    </td>
  </tr>
  <tr>
    <td></td>
    <td align="right">
     <input type="submit" value="Submit" class="button">
     </td>
  </tr>
< /table>
< /form>

< form action='tempconvert.asmx/CelsiusToFahrenheit'
method="post" target="_blank">
< table>
  <tr>
    <td>Celsius to Fahrenheit:</td>
    <td>
    <input class="frmInput" type="text" size="30" name="Celsius">
    </td>
  </tr>
  <tr>
    <td></td>
    <td align="right">
    <input type="submit" value="Submit" class="button">
    </td>
  </tr>
< /table>
< /form>
```

Substitute the "tempconvert.asmx" with the address of your web service like:

http://www.example.com/webservices/tempconvert.asmx

Introduction to WSDL

WSDL stands for Web Services Description Language.

WSDL is a language for describing web services and how to access them.

WSDL is written in XML.

What You Should Already Know

Before you continue you should have a basic understanding of the following:

XML

XML Namespaces

XML Schema

If you want to study these subjects first, find the tutorials on our Home page.

What is WSDL?

WSDL stands for Web Services Description Language

WSDL is written in XML

WSDL is an XML document

WSDL is used to describe Web services

WSDL is also used to locate Web services

WSDL is a W3C recommendation

WSDL Describes Web Services

WSDL stands for Web Services Description Language.

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.

WSDL is a W3C Recommendation

WSDL became a W3C Recommendation 26. June 2007.

WSDL Documents

The WSDL Document Structure

A WSDL document describes a web service using these major elements:

Element	Description
<types>	A container for data type definitions used by the web service
<message>	A typed definition of the data being communicated
<portType>	A set of operations supported by one or more endpoints
<binding>	A protocol and data format specification for a particular port type

WSDL Documents

The main structure of a WSDL document looks like this:

<definitions>

< types>
  data type definitions.....
< /types>

< message>
  definition of the data being communicated....
< /message>

< portType>
  set of operations.....
< /portType>

< binding>
  protocol and data format specification....
< /binding>

< /definitions>

A WSDL document can also contain other elements, like extension elements, and a service element that makes it possible to group together the definitions of several web services in one single WSDL document.

WSDL Documents

WSDL Ports

The **<portType>** element is the most important WSDL element.

It describes a web service, the operations that can be performed, and the messages that are involved.

The <portType> element can be compared to a function library (or a module, or a class) in a traditional programming language.

WSDL Messages

The **<message>** element defines the 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.

WSDL Types

The **<types>** element defines the data types that are used by the web service.

For maximum platform neutrality, WSDL uses XML Schema syntax to define data types.

WSDL Bindings

The **<binding>** element defines the data format and protocol for each port type.

WSDL Documents

WSDL Example

This is a simplified fraction of a WSDL document:

```
<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>
```

In this example the **<portType>** element defines "glossaryTerms" as the name of a **port**, and "getTerm" as the name of an **operation**.

The "getTerm" operation has an **input message** called "getTermRequest" and an **output message** called "getTermResponse".

The **<message>** elements define the **parts** of each message and the associated data types.

Compared to traditional programming, glossaryTerms is a function library, "getTerm" is a function with "getTermRequest" as the input parameter, and getTermResponse as the return parameter.

WSDL PortType

The **<portType>** element is the most important WSDL element.

WSDL - The <portType> Element

The <portType> element defines **a web service**, the **operations** that can be performed, and the **messages** that are involved.

<portType> defines the connection point to a web service. It can be compared to a function library (or a module, or a class) in a traditional programming language. Each operation can be compared to a function in a traditional programming language.

WSDL PortType

Operation Types
The request-response type is the most common operation type, but WSDL defines four types:

Type	Definition
One-way	The operation can receive a message but will not return a response
Request-response	The operation can receive a request and will return a response
Solicit-response	The operation can send a request and will wait for a response
Notification	The operation can send a message but will not wait for a response

WSDL PortType

One-Way Operation

A one-way operation example:

```
<message name="newTermValues">
  <part name="term" type="xs:string"/>
  <part name="value" type="xs:string"/>
< /message>

< portType name="glossaryTerms">
  <operation name="setTerm">
    <input name="newTerm" message="newTermValues"/>
  </operation>
< /portType >
```

In the example above, the portType "glossaryTerms" defines a one-way operation called "setTerm".

The "setTerm" operation allows input of new glossary terms messages using a "newTermValues" message with the input parameters "term" and "value". However, no output is defined for the operation.

WSDL PortType

Request-Response Operation

A request-response operation 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>
```

In the example above, the portType "glossaryTerms" defines a request-response operation called "getTerm".

The "getTerm" operation requires an input message called "getTermRequest" with a parameter called "term", and will return an output message called "getTermResponse" with a parameter called "value".

WSDL Bindings

WSDL bindings defines the message format and protocol details for a web service.

Binding to SOAP

A request-response operation 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>
```

WSDL Bindings

The **binding** element has two attributes - name and type.

The name attribute (you can use any name you want) defines the name of the binding, and the type attribute points to the port for the binding, in this case the "glossaryTerms" port.

The **soap:binding** element has two attributes - style and transport.

The style attribute can be "rpc" or "document". In this case we use document. The transport attribute defines the SOAP protocol to use. In this case we use HTTP.

The **operation** element defines each operation that the portType exposes.

For each operation the corresponding SOAP action has to be defined. You must also specify how the input and output are encoded. In this case we use "literal".

WSDL and UDDI

Universal Description, Discovery and Integration (UDDI) is a directory service where businesses can register and search for Web services.

What is UDDI

UDDI is a platform-independent framework for describing services, discovering businesses, and integrating business services by using the Internet.

UDDI stands for Universal Description, Discovery and Integration

UDDI is a directory for storing information about web services

UDDI is a directory of web service interfaces described by WSDL

UDDI communicates via SOAP

UDDI is built into the Microsoft .NET platform

What is UDDI Based On?

UDDI uses World Wide Web Consortium (W3C) and Internet Engineering Task Force (IETF) Internet standards such as XML, HTTP, and DNS protocols.

UDDI uses WSDL to describe interfaces to web services

Additionally, cross platform programming features are addressed by adopting SOAP, known as XML Protocol messaging specifications found at the W3C Web site.

WSDL and UDDI

UDDI Benefits

Any industry or businesses of all sizes can benefit from UDDI.

Before UDDI, there was no Internet standard for businesses to reach their customers and partners with information about their products and services. Nor was there a method of how to integrate into each other's systems and processes.

Problems the UDDI specification can help to solve:

- Making it possible to discover the right business from the millions currently online
- Defining how to enable commerce once the preferred business is discovered
- Reaching new customers and increasing access to current customers
- Expanding offerings and extending market reach
- Solving customer-driven need to remove barriers to allow for rapid participation in the global Internet economy
- Describing services and business processes programmatically in a single, open, and secure environment

How can UDDI be Used

If the industry published an UDDI standard for flight rate checking and reservation, airlines could register their services into an UDDI directory. Travel agencies could then search the UDDI directory to find the airline's reservation interface. When the interface is found, the travel agency can communicate with the service immediately because it uses a well-defined reservation interface.

Who is Supporting UDDI?

UDDI is a cross-industry effort driven by all major platform and software providers like Dell, Fujitsu, HP, Hitachi, IBM, Intel, Microsoft, Oracle, SAP, and Sun, as well as a large community of marketplace operators, and e-business leaders.

Over 220 companies are members of the UDDI community.

SOAP Introduction

SOAP stands for Simple Object Access Protocol.

SOAP is a protocol for accessing web services.

SOAP is based on XML.

What You Should Already Know

Before you study SOAP you should have a basic understanding of XML and XML Namespaces.

If you want to study these subjects first, please read our XML Tutorial.

What is SOAP?

SOAP stands for Simple Object Access Protocol

SOAP is a communication protocol

SOAP is for communication between applications

SOAP is a format for sending messages

SOAP communicates via Internet

SOAP is platform independent

SOAP is language independent

SOAP is based on XML

SOAP is simple and extensible

SOAP allows you to get around firewalls

SOAP is a W3C recommendation

SOAP Introduction

Why SOAP?

It is important for application development to allow Internet communication between programs.

Today's applications communicate using Remote Procedure Calls (RPC) between objects like DCOM and CORBA, but HTTP was not designed for this. RPC represents a compatibility and security problem; firewalls and proxy servers will normally block this kind of traffic.

A better way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this.

SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.

SOAP is a W3C Recommendation

SOAP became a W3C Recommendation 24. June 2003.

SOAP Syntax

SOAP Building Blocks

A SOAP message is an ordinary XML document containing the following elements:

An Envelope element that identifies the XML document as a SOAP message

A Header element that contains header information

A Body element that contains call and response information

A Fault element containing errors and status information

All the elements above are declared in the default namespace for the SOAP envelope:

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

and the default namespace for SOAP encoding and data types is:

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

Syntax Rules

Here are some important 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

SOAP Syntax

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>
```

SOAP Envelope Element

The SOAP Envelope element is the root element of a SOAP message.

The SOAP Envelope Element

The required SOAP Envelope element is the root element of a SOAP message. This element defines the XML document as a SOAP message.

Example

```
<?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>
```

The xmlns:soap Namespace

Notice the xmlns:soap namespace in the example above. It should always have the value of: "http://www.w3.org/2001/12/soap-envelope".

The namespace defines the Envelope as a SOAP Envelope.

If a different namespace is used, the application generates an error and discards the message.

SOAP Envelope Element

The encodingStyle Attribute

The encodingStyle attribute is used to define the data types used in the document. This attribute may appear on any SOAP element, and applies to the element's contents and all child elements.

A SOAP message has no default encoding.

Syntax

soap:encodingStyle="*URI*"

Example

```
<?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>
```

SOAP Header Element

The SOAP Header element contains header information.

The SOAP Header Element

The optional SOAP Header element contains application-specific information (like authentication, payment, etc) about the SOAP message.

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.

```
<?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>
```

SOAP Header Element

The example above contains a header with a "Trans" element, a "mustUnderstand" attribute with a value of 1, and a value of 234.

SOAP defines three attributes in the default namespace ("http://www.w3.org/2001/12/soap-envelope"). These attributes are: mustUnderstand, actor, and encodingStyle.

The attributes defined in the SOAP Header defines how a recipient should process the SOAP message.

The mustUnderstand Attribute

The SOAP mustUnderstand attribute can be used to indicate whether a header entry is mandatory or optional for the recipient to process.

If you add mustUnderstand="1" to a child element of the Header element it indicates that the receiver processing the Header must recognize the element. If the receiver does not recognize the element it will fail when processing the Header.

Syntax

soap:mustUnderstand="0|1"

SOAP Header Element

Example

```
<?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>
```

SOAP Header Element

The actor Attribute

A SOAP message may travel from a sender to a receiver by passing different endpoints along the message path. However, not all parts of a SOAP message may be intended for the ultimate endpoint, instead, it may be intended for one or more of the endpoints on the message path.

The SOAP actor attribute is used to address the Header element to a specific endpoint.

Syntax

soap:actor="*URI*"

Example

```
<?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:actor="http://www.w3schools.com/appml/">234
 < /m:Trans>
< /soap:Header>
...
...
</soap:Envelope>
```

The encodingStyle Attribute

The encodingStyle attribute is 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*"

SOAP Body Element

The SOAP Body element contains the actual SOAP message.

The SOAP Body Element

The required SOAP Body element contains the actual SOAP message intended for the ultimate endpoint of the message.

Immediate child elements of the SOAP Body element may be namespace-qualified.

Example

```
<?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>
```

SOAP Body Element

The example above requests the price of apples. Note that the m:GetPrice and the Item elements above are application-specific elements. They are not a part of the SOAP namespace.

A SOAP response could look something like this:

```
<?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 Fault Element

The SOAP Fault element holds errors and status information for a SOAP message.

The SOAP Fault Element

The optional SOAP Fault element is used to indicate error messages.

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.

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 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

SOAP HTTP Binding

The HTTP Protocol

HTTP communicates over TCP/IP. An HTTP client connects to an HTTP server using TCP. After establishing a connection, the client can send an HTTP request message to the server:

POST /item HTTP/1.1
Host: 189.123.255.239
Content-Type: text/plain
Content-Length: 200

The server then processes the request and sends an HTTP response back to the client. The response contains a status code that indicates the status of the request:

200 OK
Content-Type: text/plain
Content-Length: 200

In the example above, the server returned a status code of 200. This is the standard success code for HTTP.

If the server could not decode the request, it could have returned something like this:

400 Bad Request
Content-Length: 0

SOAP HTTP Binding

SOAP HTTP Binding

A SOAP method is an HTTP request/response that complies with the SOAP encoding rules.

HTTP + XML = SOAP

A SOAP request could be an HTTP POST or an HTTP GET request.

The HTTP POST request specifies at least two HTTP headers: Content-Type and Content-Length.

Content-Type

The Content-Type header for a SOAP request and response defines the MIME type for the message and the character encoding (optional) used for the XML body of the request or response.

Syntax

Content-Type: MIMEType; charset=character-encoding

Example

POST /item HTTP/1.1
Content-Type: application/soap+xml; charset=utf-8

Content-Length

The Content-Length header for a SOAP request and response specifies the number of bytes in the body of the request or response.

Syntax

Content-Length: bytes

Example

POST /item HTTP/1.1
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 250

SOAP Example

In the example below, a GetStockPrice request is sent to a server. The request has a StockName parameter, and a Price parameter that will be returned in the response. The namespace for the function is defined in "http://www.example.org/stock".

A SOAP request:

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

< ?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 xmlns:m="http://www.example.org/stock">
  <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>

< /soap:Envelope>
```

SOAP Example

The SOAP response:

HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

```
< ?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 xmlns:m="http://www.example.org/stock">
  <m:GetStockPriceResponse>
    <m:Price>34.5</m:Price>
  </m:GetStockPriceResponse>
</soap:Body>

< /soap:Envelope>
```

Introduction to RDF

RDF stands for Resource Description Framework.

RDF is a standard for describing Web resources.

RDF can be used to describe title, author, content, and copyright information of web pages.

RDF Document Example

```
<?xml version="1.0"?>

< rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:si="http://www.w3schools.com/rdf/">

<rdf:Description rdf:about="http://www.w3schools.com">
  <si:title>W3Schools</si:title>
  <si:author>Jan Egil Refsnes</si:author>
</rdf:Description>

< /rdf:RDF>
```

What You Should Already Know

Before you continue you should have a basic understanding of the following:

HTML

XML and XML Namespaces

If you want to study these subjects first, find the tutorials on our Home page.

Introduction to RDF

What is RDF?

RDF stands for **R**esource **D**escription **F**ramework

RDF is a framework for describing resources on the web

RDF is designed to be read and understood by computers

RDF is not designed for being displayed to people

RDF is written in XML

RDF is a part of the W3C's Semantic Web Activity

RDF is a W3C Recommendation

RDF - Examples of Use

Describing properties for shopping items, such as price and availability

Describing time schedules for web events

Describing information about web pages (content, author, created and modified date)

Describing content and rating for web pictures

Describing content for search engines

Describing electronic libraries

RDF is Designed to be Read by Computers

RDF was designed to provide a common way to describe information so it can be read and understood by computer applications.

RDF descriptions are not designed to be displayed on the web.

Introduction to RDF

RDF is Written in XML

RDF documents are written in XML. The XML language used by RDF is called RDF/XML.

By using XML, RDF information can easily be exchanged between different types of computers using different types of operating systems and application languages.

RDF and "The Semantic Web"

The RDF language is a part of the W3C's Semantic Web Activity. W3C's "Semantic Web Vision" is a future where:

Web information has exact meaning

Web information can be understood and processed by computers

Computers can integrate information from the web

RDF is a W3C Recommendation

RDF became a W3C Recommendation 10. February 2004.

RDF Rules

RDF uses Web identifiers (URIs) to identify resources.

RDF describes resources with properties and property values.

RDF Resource, Property, and Property Value

RDF identifies things using Web identifiers (URIs), and describes resources with properties and property values.

Explanation of Resource, Property, and Property value:

A **Resource** is anything that can have a URI, such as "http://www.w3schools.com/rdf"

A **Property** is a Resource that has a name, such as "author" or "homepage"

A **Property value** is the value of a Property, such as "Jan Egil Refsnes" or "http://www.w3schools.com" (note that a property value can be another resource)

The following RDF document could describe the resource "http://www.w3schools.com/rdf":

```
<?xml version="1.0"?>

< RDF>
  <Description about="http://www.w3schools.com/rdf">
    <author>Jan Egil Refsnes</author>
    <homepage>http://www.w3schools.com</homepage>
  </Description>
< /RDF>
```

The example above is simplified. Namespaces are omitted.

RDF Rules

RDF Statements

The combination of a Resource, a Property, and a Property value forms a **Statement** (known as the **subject, predicate and object** of a Statement).

Let's look at some example statements to get a better understanding:

Statement: "The author of http://www.w3schools.com/rdf is Jan Egil Refsnes".

The subject of the statement above is: http://www.w3schools.com/rdf

The predicate is: author

The object is: Jan Egil Refsnes

Statement: "The homepage of http://www.w3schools.com/rdf is http://www.w3schools.com".

The subject of the statement above is: http://www.w3schools.com/rdf

The predicate is: homepage

The object is: http://www.w3schools.com

RDF Example

Here are two records from a CD-list:

Title	Artist	Country	Company	Price	Year
Empire Burlesque	Bob Dylan	USA	Columbia	١٠,٩٠	١٩٨٥
Hide your heart	Bonnie Tyler	UK	CBS Records	٩,٩٠	١٩٨٨

RDF Example

Below is a few lines from an RDF document:

```
<?xml version="1.0"?>

< rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.recshop.fake/cd#">

< rdf:Description
rdf:about="http://www.recshop.fake/cd/Empire Burlesque">
  <cd:artist>Bob Dylan</cd:artist>
  <cd:country>USA</cd:country>
  <cd:company>Columbia</cd:company>
  <cd:price>10.90</cd:price>
  <cd:year>1985</cd:year>
< /rdf:Description>

< rdf:Description
rdf:about="http://www.recshop.fake/cd/Hide your heart">
  <cd:artist>Bonnie Tyler</cd:artist>
  <cd:country>UK</cd:country>
  <cd:company>CBS Records</cd:company>
  <cd:price>9.90</cd:price>
  <cd:year>1988</cd:year>
< /rdf:Description>
```

RDF Example

The first line of the RDF document is the XML declaration. The XML declaration is followed by the root element of RDF documents: **<rdf:RDF>**.

The **xmlns:rdf** namespace, specifies that elements with the rdf prefix are from the namespace "http://www.w3.org/1999/02/22-rdf-syntax-ns#".

The **xmlns:cd** namespace, specifies that elements with the cd prefix are from the namespace "http://www.recshop.fake/cd#".

The **<rdf:Description>** element contains the description of the resource identified by the **rdf:about** attribute.

The elements: **<cd:artist>, <cd:country>, <cd:company>,** etc. are properties of the resource.

RDF Example

RDF Online Validator

W3C's RDF Validation Service is useful when learning RDF. Here you can experiment with RDF files.

The online RDF Validator parses your RDF document, checks your syntax, and generates tabular and graphical views of your RDF document.

Copy and paste the example below into W3C's RDF validator:

```
<?xml version="1.0"?>

< rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:si="http://www.w3schools.com/rdf/">
<rdf:Description rdf:about="http://www.w3schools.com">
  <si:title>W3Schools.com</si:title>
  <si:author>Jan Egil Refsnes</si:author>
</rdf:Description>
< /rdf:RDF>
```

When you parse the example above, the result will look something like this.

RDF Main Elements

The main elements of RDF are the root element, <RDF>, and the <Description> element, which identifies a resource.

The <rdf:RDF> Element

<rdf:RDF> is the root element of an RDF document. It defines the XML document to be an RDF document. It also contains a reference to the RDF namespace:

```
<?xml version="1.0"?>

< rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  ...Description goes here...
< /rdf:RDF>
```

RDF Main Elements

The <rdf:Description> Element

The <rdf:Description> element identifies a resource with the about attribute.

The <rdf:Description> element contains elements that describe the resource:

```
<?xml version="1.0"?>

< rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.recshop.fake/cd#">

< rdf:Description
rdf:about="http://www.recshop.fake/cd/Empire Burlesque">
  <cd:artist>Bob Dylan</cd:artist>
  <cd:country>USA</cd:country>
  <cd:company>Columbia</cd:company>
  <cd:price>10.90</cd:price>
  <cd:year>1985</cd:year>
< /rdf:Description>

< /rdf:RDF>
```

The elements, artist, country, company, price, and year, are defined in the http://www.recshop.fake/cd# namespace. This namespace is outside RDF (and not a part of RDF). RDF defines only the framework. The elements, artist, country, company, price, and year, must be defined by someone else (company, organization, person, etc).

RDF Main Elements

Properties as Attributes

The property elements can also be defined as attributes (instead of elements):

```
<?xml version="1.0"?>

< rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.recshop.fake/cd#">

< rdf:Description
rdf:about="http://www.recshop.fake/cd/Empire Burlesque"
cd:artist="Bob Dylan" cd:country="USA"
cd:company="Columbia" cd:price="10.90"
cd:year="1985" />

< /rdf:RDF>
```

RDF Main Elements

Properties as Resources

The property elements can also be defined as resources:

```
<?xml version="1.0"?>

< rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.recshop.fake/cd#">

< rdf:Description
rdf:about="http://www.recshop.fake/cd/Empire Burlesque">
  <cd:artist rdf:resource="http://www.recshop.fake/cd/dylan" />
  ...
  ...
< /rdf:Description>

< /rdf:RDF>
```

In the example above, the property artist does not have a value, but a reference to a resource containing information about the artist.

RDF Container Elements

RDF containers are used to describe group of things.

The following RDF elements are used to describe groups: <Bag>, <Seq>, and <Alt>.

The <rdf:Bag> Element

The <rdf:Bag> element is used to describe a list of values that do not have to be in a specific order.

The <rdf:Bag> element may contain duplicate values.

Example

```
<?xml version="1.0"?>
< rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.recshop.fake/cd#">
< rdf:Description
rdf:about="http://www.recshop.fake/cd/Beatles">
  <cd:artist>
    <rdf:Bag>
      <rdf:li>John</rdf:li>
      <rdf:li>Paul</rdf:li>
      <rdf:li>George</rdf:li>
      <rdf:li>Ringo</rdf:li>
    </rdf:Bag>
  </cd:artist>
< /rdf:Description>
< /rdf:RDF>
```

RDF Container Elements

The <rdf:Seq> Element

The <rdf:Seq> element is used to describe an ordered list of values (For example, in alphabetical order).

The <rdf:Seq> element may contain duplicate values.

Example

```
<?xml version="1.0"?>

< rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.recshop.fake/cd#">

< rdf:Description
rdf:about="http://www.recshop.fake/cd/Beatles">
  <cd:artist>
    <rdf:Seq>
      <rdf:li>George</rdf:li>
      <rdf:li>John</rdf:li>
      <rdf:li>Paul</rdf:li>
      <rdf:li>Ringo</rdf:li>
    </rdf:Seq>
  </cd:artist>
< /rdf:Description>

< /rdf:RDF>
```

RDF Container Elements

The <rdf:Alt> Element

The <rdf:Alt> element is used to describe a list of alternative values (the user can select only one of the values).

Example

```
<?xml version="1.0"?>

< rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.recshop.fake/cd#">

< rdf:Description
rdf:about="http://www.recshop.fake/cd/Beatles">
  <cd:format>
    <rdf:Alt>
      <rdf:li>CD</rdf:li>
      <rdf:li>Record</rdf:li>
      <rdf:li>Tape</rdf:li>
    </rdf:Alt>
  </cd:format>
< /rdf:Description>

< /rdf:RDF>
```

RDF Container Elements

RDF Terms

In the examples above we have talked about "list of values" when describing the container elements. In RDF these "list of values" are called members.

So, we have the following:

A container is a resource that contains things

The contained things are called members (not list of values)

RDF Collections

RDF collections describe groups that can ONLY contain the specified members.

The rdf:parseType="Collection" Attribute

As seen in the previous chapter, a container says that the containing resources are members - it does not say that other members are not allowed.

RDF collections are used to describe groups that can ONLY contain the specified members.

A collection is described by the attribute rdf:parseType="Collection".

Example

```
<?xml version="1.0"?>
< rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://recshop.fake/cd#">

< rdf:Description
rdf:about="http://recshop.fake/cd/Beatles">
 < cd:artist rdf:parseType="Collection">
   < rdf:Description rdf:about="http://recshop.fake/cd/Beatles/George"/>
   < rdf:Description rdf:about="http://recshop.fake/cd/Beatles/John"/>
   < rdf:Description rdf:about="http://recshop.fake/cd/Beatles/Paul"/>
   < rdf:Description rdf:about="http://recshop.fake/cd/Beatles/Ringo"/>
 < /cd:artist>
< /rdf:Description>

< /rdf:RDF>
```

RDF Schema (RDFS)

RDF Schema (RDFS) is an extension to RDF.

RDF Schema and Application Classes

RDF describes resources with classes, properties, and values.

In addition, RDF also needs a way to define application-specific classes and properties. Application-specific classes and properties must be defined using extensions to RDF.

One such extension is RDF Schema.

RDF Schema (RDFS)

RDF Schema does not provide actual application-specific classes and properties.

Instead RDF Schema provides the framework to describe application-specific classes and properties.

Classes in RDF Schema are much like classes in object oriented programming languages. This allows resources to be defined as instances of classes, and subclasses of classes.

RDF Schema (RDFS)

RDFS Example

The following example demonstrates some of the RDFS facilities:

```
<?xml version="1.0"?>

< rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xml:base="http://www.animals.fake/animals#">

< rdf:Description rdf:ID="animal">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
< /rdf:Description>

< rdf:Description rdf:ID="horse">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#animal"/>
< /rdf:Description>

< /rdf:RDF>
```

In the example above, the resource "horse" is a subclass of the class "animal".

RDF Schema (RDFS)

Example Abbreviated

Since an RDFS class is an RDF resource we can abbreviate the example above by using rdfs:Class instead of rdf:Description, and drop the rdf:type information:

```
<?xml version="1.0"?>

< rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xml:base="http://www.animals.fake/animals#">

< rdfs:Class rdf:ID="animal" />

< rdfs:Class rdf:ID="horse">
  <rdfs:subClassOf rdf:resource="#animal"/>
< /rdfs:Class>

< /rdf:RDF>
```

That's it!

RDF Dublin Core Metadata Initiative

The Dublin Core Metadata Initiative (DCMI) has created some predefined properties for describing documents.

The Dublin Core

RDF is metadata (data about data). RDF is used to describe information resources.

The Dublin Core is a set of predefined properties for describing documents.

The first Dublin Core properties were defined at the **Metadata Workshop in Dublin, Ohio** in 1995 and is currently maintained by the Dublin Core Metadata Initiative.

RDF Dublin Core Metadata Initiative

Property	Definition
Contributor	An entity responsible for making contributions to the content of the resource
Coverage	The extent or scope of the content of the resource
Creator	An entity primarily responsible for making the content of the resource
Format	The physical or digital manifestation of the resource
Date	A date of an event in the lifecycle of the resource
Description	An account of the content of the resource
Identifier	An unambiguous reference to the resource within a given context
Language	A language of the intellectual content of the resource
Publisher	An entity responsible for making the resource available
Relation	A reference to a related resource
Rights	Information about rights held in and over the resource
Source	A Reference to a resource from which the present resource is derived
Subject	A topic of the content of the resource
Title	A name given to the resource
Type	The nature or genre of the content of the resource

A quick look at the table above indicates that RDF is ideal for representing Dublin Core information.

RDF Dublin Core Metadata Initiative

RDF Example

The following example demonstrates the use of some of the Dublin Core properties in an RDF document:

```
<?xml version="1.0"?>

< rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc= "http://purl.org/dc/elements/1.1/">

< rdf:Description rdf:about="http://www.w3schools.com">
  <dc:description>W3Schools - Free tutorials</dc:description>
  <dc:publisher>Refsnes Data as</dc:publisher>
  <dc:date>2008-09-01</dc:date>
  <dc:type>Web Development</dc:type>
  <dc:format>text/html</dc:format>
  <dc:language>en</dc:language>
< /rdf:Description>

< /rdf:RDF>
```

Introduction to OWL

OWL is a language for processing web information.

What is OWL?

OWL stands for Web Ontology Language

OWL is built on top of RDF

OWL is for processing information on the web

OWL was designed to be interpreted by computers

OWL was not designed for being read by people

OWL is written in XML

OWL has three sublanguages

OWL is a W3C standard

What is Ontology?

Ontology is about the exact description of things and their relationships.

For the web, ontology is about the exact description of web information and relationships between web information.

Introduction to OWL

Why OWL?

OWL is a part of the "Semantic Web Vision" - a future where:

Web information has exact meaning

Web information can be processed by computers

Computers can integrate information from the web

OWL was Designed for Processing Information

OWL was designed to provide a common way to process the content of web information (instead of displaying it).

OWL was designed to be read by computer applications (instead of humans).

OWL is Different from RDF

OWL and RDF are much of the same thing, but OWL is a stronger language with greater machine interpretability than RDF.

OWL comes with a larger vocabulary and stronger syntax than RDF.

Introduction to OWL

OWL Sublanguages

OWL has three sublanguages:

OWL Lite

OWL DL (includes OWL Lite)

OWL Full (includes OWL DL)

OWL is Written in XML

By using XML, OWL information can easily be exchanged between different types of computers using different types of operating system and application languages.

OWL is a Web Standard

OWL became a W3C (World Wide Web Consortium) Recommendation in February 2004.

A W3C Recommendation is understood by the industry and the web community as a web standard. A W3C Recommendation is a stable specification developed by a W3C Working Group and reviewed by the W3C Membership.

RDF Reference

The RDF Namespaces

The RDF namespace (xmlns:rdf) is:
http://www.w3.org/1999/02/22-rdf-syntax-ns#

The RDFS namespace (xmlns:rdfs) is:
http://www.w3.org/2000/01/rdf-schema#

The RDF Extension and MIME Type

The recommended file extension for RDF files is **.rdf**. However, the extension **.xml** is often used to provide compatibility with old xml parsers.

The MIME type should be **"application/rdf+xml"**.

RDF Reference

- **RDFS / RDF Classes**

Element	Class of	Subclass of
rdfs:Class	All classes	
rdfs:Datatype	Data types	Class
rdfs:Resource	All resources	Class
rdfs:Container	Containers	Resource
rdfs:Literal	Literal values (text and numbers)	Resource
rdf:List	Lists	Resource
rdf:Property	Properties	Resource
rdf:Statement	Statements	Resource
rdf:Alt	Containers of alternatives	Container
rdf:Bag	Unordered containers	Container
rdf:Seq	Ordered containers	Container
rdfs:ContainerMembershipProperty	Container membership properties	Property
rdf:XMLLiteral	XML literal values	Literal

RDF Reference
RDFS / RDF Properties

Element	Domain	Range	Description
rdfs:domain	Property	Class	The domain of the resource
rdfs:range	Property	Class	The range of the resource
rdfs:subPropertyOf	Property	Property	The property is a sub property of a property
rdfs:subClassOf	Class	Class	The resource is a subclass of a class
rdfs:comment	Resource	Literal	The human readable description of the resource
rdfs:label	Resource	Literal	The human readable label (name) of the resource
rdfs:isDefinedBy	Resource	Resource	The definition of the resource
rdfs:seeAlso	Resource	Resource	The additional information about the resource
rdfs:member	Resource	Resource	The member of the resource
rdf:first	List	Resource	
rdf:rest	List	List	
rdf:subject	Statement	Resource	The subject of the resource in an RDF Statement
rdf:predicate	Statement	Resource	The predicate of the resource in an RDF Statement
rdf:object	Statement	Resource	The object of the resource in an RDF Statement
rdf:value	Resource	Resource	The property used for values
rdf:type	Resource	Class	The resource is an instance of a class

RDF Attributes
RDF Reference

Element	Domain	Range	Description
rdf:about			Defines the resource being described
rdf:Description			Container for the description of a resource
rdf:resource			Defines a resource to identify a property
rdf:datatype			Defines the data type of an element
rdf:ID			Defines the ID of an element
rdf:li			Defines a list
rdf:_n			Defines a node
rdf:nodeID			Defines the ID of an element node
rdf:parseType			Defines how an element should be parsed
rdf:RDF			The root of an RDF document
xml:base			Defines the XML base
xml:lang			Defines the language of the element content
rdf:aboutEach			(removed)
rdf:aboutEachPrefix			(removed)
rdf:bagID			(removed)

Elements described as (removed) are removed from the latest RDF standard.