Web Services

- A web service is any piece of software that makes itself available over the internet and uses a standardized XML messaging system.
- client server application or application component for communication between two devices over network.
- The main purpose of web services is to exchange data.
- Web services communicate using open protocols.
- As all communication is in XML, web services are not tied to any one operating system or programming language--Java can talk with Perl; Windows applications can talk with Unix applications.

Architecture

A Web service is a software system designed to support interoperable machine-to-machine interaction over a network.

There are three major roles within the web service architecture:

- **Service Provider**

  This is the provider of the web service. The service provider implements the service and makes it available on the Internet.

- **Service Requestor**

  This is any consumer of the web service. The requestor utilizes an existing web service by opening a network connection and sending an XML request.

- **Service Registry**

  This is a logically centralized directory of services. The registry provides a central place where developers can publish new services or find existing ones. It therefore serves as a centralized clearing house for companies and their services.

Architecture

Web Service Protocol Stack

The web service protocol stack currently has four main layers.

- **Service Transport**

  This layer is responsible for transporting messages between applications. Currently, this layer includes Hyper Text Transport Protocol (HTTP), Simple Mail Transfer Protocol (SMTP), File Transfer Protocol (FTP), and newer protocols such as Blocks Extensible Exchange Protocol (BEEP).

- **XML Messaging**

  This layer is responsible for encoding messages in a common XML format so that messages can be understood at either end. Currently, this layer includes XML-RPC and SOAP.

- **Service Description**

  This layer is responsible for describing the public interface to a specific web service. Currently, service description is handled via the Web Service Description Language (WSDL).

- **Service Discovery**

  This layer is responsible for centralizing services into a common registry and providing easy publish/find functionality. Currently, service discovery is handled via Universal Description, Discovery, and Integration (UDDI).

Components of web services

- **XML**- Describes only data. So, any application that understands XML-regardless of the application's programming language or platform-has the ability to format XML in a variety of ways (well-formed or valid).
- **SOAP(Simple Object Access Protocol)**- Provides a communication mechanism between services and applications.
- **WSDL(Web Services Description Language)**- Offers a uniform method of describing web services to other programs.
- **UDDI(Universal Description, Discovery and Integration)**- Enables the creation of searchable Web services registries.

SOAP

- XML-based messaging protocol for exchanging information among computers.
- communication protocol designed to communicate via Internet.
- can extend HTTP for XML messaging.
- provides data transport for Web services.
- SOAP commonly uses HTTP, but other protocols such as Simple Mail Transfer Protocol (SMTP) may be used.
- can exchange complete documents or call a remote procedure.
- can be used for broadcasting a message.
- is platform and language-independent.
- is the XML way of defining what information is sent and how.
- enables client applications to easily connect to remote services and invoke remote methods.

SOAP

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

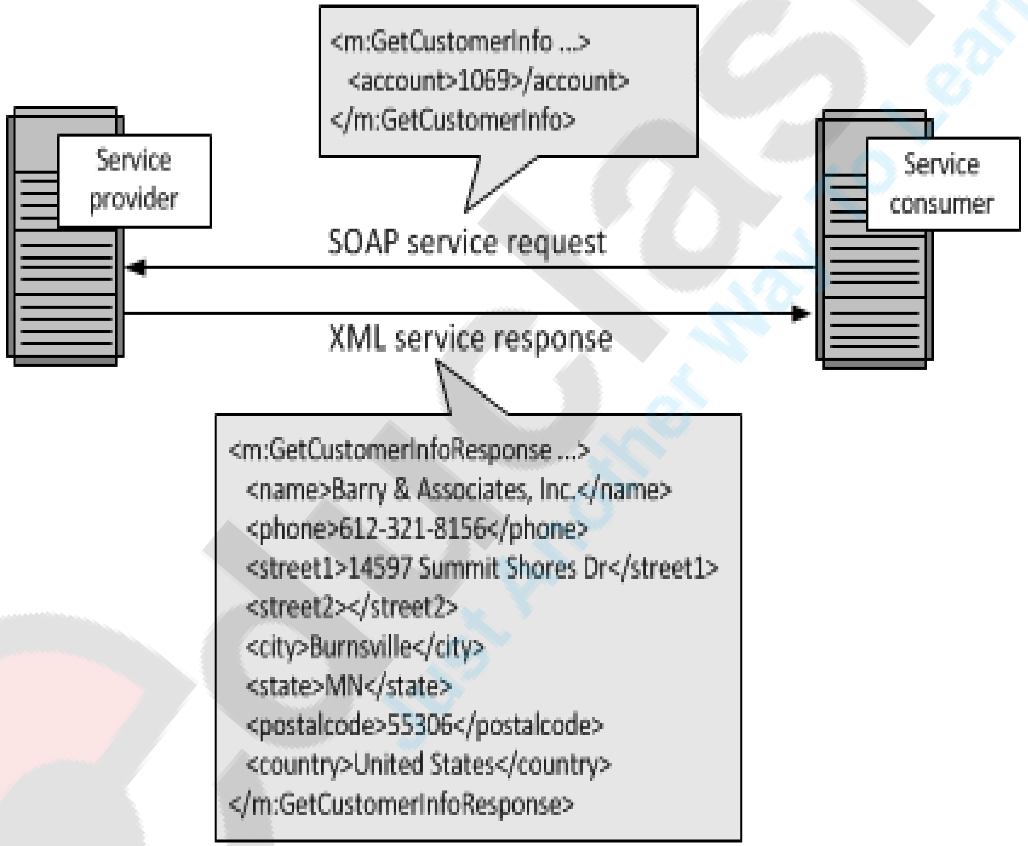
- **Envelope** – Defines the start and the end of the message. It is a mandatory element.
- **Header** – Contains any optional attributes of the message used in processing the message, either at an intermediary point or at the ultimate end-point. It is an optional element.
- **Body** – Contains the XML data comprising the message being sent. It is a mandatory element.
- **Fault** – An optional Fault element that provides information about errors that occur while processing the message.

The following block depicts the general structure of a SOAP message –

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
  xmlns="http://schemas.xmlsoap.org/soap/envelope/">

    < SOAP-ENV:Header>
      ...
    </ SOAP-ENV:Header>

    < SOAP-ENV:Body>
       ...
    </ SOAP-ENV:Body>
   ...
</ SOAP-ENV: Envelope>
```



```
<m:GetCustomerInfo ...>
  <account>1069>/account>
</m:GetCustomerInfo>
```

```
<m:GetCustomerInfoResponse ...>
  <name>Barry & Associates, Inc.</name>
  <phone>612-321-8156</phone>
  <street1>14597 Summit Shores Dr</street1>
  <street2></street2>
  <city>Burnsville</city>
  <state>MN</state>
  <postalcode>55306</postalcode>
  <country>United States</country>
</m:GetCustomerInfoResponse>
```

SOAP

- The **SOAP envelope** indicates the start and the end of the message so that the receiver knows when an entire message has been received.
- The SOAP envelope is therefore basically a packaging mechanism.
- Every SOAP message has a root Envelope element.
- Envelope is a mandatory part of SOAP message.
- Every Envelope element must contain exactly one Body element.
- If an Envelope contains a Header element, it must contain no more than one, and it must appear as the first child of the Envelope, before the Body.
- The envelope changes when SOAP versions change.
- The SOAP envelope is specified using the *ENV* namespace prefix and the Envelope element.
- The optional SOAP encoding is also specified using a namespace name and the optional *encodingStyle* element, which could also point to an encoding style other than the SOAP one.

SOAP

- The optional Header element offers a flexible framework for specifying additional application-level requirements.
- It is an optional part of a SOAP message.
- Header elements can occur multiple times.
- Headers are intended to add new features and functionality.
- The SOAP header contains header entries defined in a namespace.
- The header is encoded as the first immediate child element of the SOAP envelope.
- When multiple headers are defined, all immediate child elements of the SOAP header are interpreted as SOAP header blocks.

SOAP

A SOAP Header can have the following two attributes –

- **Actor attribute**

The SOAP protocol defines a message path as a list of SOAP service nodes. Each of these intermediate nodes can perform some processing and then forward the message to the next node in the chain. By setting the Actor attribute, the client can specify the recipient of the SOAP header.

- **MustUnderstand attribute**

It indicates whether a Header element is optional or mandatory. If set to true, the recipient must understand and process the Header attribute according to its defined semantics, or return a fault.

SOAP

- The SOAP body is a mandatory element that contains the application-defined XML data being exchanged in the SOAP message.
- The body contains mandatory information intended for the ultimate receiver of the message.

For example:-

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
  xmlns="http://schemas.xmlsoap.org/soap/envelope/">

    < SOAP-ENV:Header>
     ...
    </ SOAP-ENV:Header>

    < SOAP-ENV:Body>
         <m:GetQuotation xmlns:m="http://www.tp.com/Quotation">
          <m:Item>Computers</m:Item>
         </m:GetQuotation>
    </ SOAP-ENV:Body>
   ...
</ SOAP-ENV: Envelope>
```

The above example requests for a quotation for computer sets.

SOAP

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
  xmlns="http://schemas.xmlsoap.org/soap/envelope/">

    < SOAP-ENV:Header>
     ...
    </ SOAP-ENV:Header>

    < SOAP-ENV:Body>
<m:GetQuotationResponse xmlns:m="http://www.tp.com/Quotation">
         <m:Quotation>This is Quotation</m:Quotation>
      </m:GetQuotationResponse>
</ SOAP-ENV:Body>
   ...
</ SOAP-ENV: Envelope>
```

This is the response to the above query.

SOAP

- The SOAP fault mechanism returns specific information about the error, including a predefined code, a description, and the address of the SOAP processor that generated the fault.
- A SOAP message can carry only one fault block.
- Fault is an optional part of a SOAP message.
- For HTTP binding, a successful response is linked to the 200 to 299 range of status codes.
- SOAP Fault is linked to the 500 to 599 range of status codes.

Sub-element	Description
<faultCode>	It is a text code used to indicate a class of errors.
<faultString>	It is a text message explaining the error.
<faultActor>	It is a text string indicating who caused the fault. It is useful if the SOAP message travels through several nodes in the SOAP message path, and the client needs to know which node caused the error. A node that does not act as the ultimate destination must include a *faultActor* element.
<detail>	It is an element used to carry application-specific error messages. The detail element can contain child elements called detail entries.

SOAP

SOAP includes a built-in set of rules for encoding data types. It enables the SOAP message to indicate specific data types, such as integers, floats, doubles, or arrays.

- SOAP data types are divided into two broad categories – scalar types and compound types.
- Scalar types contain exactly one value such as a last name, price, or product description.
- Compound types contain multiple values such as a purchase order or a list of stock quotes.
- Compound types are further subdivided into arrays and structs.
- The encoding style for a SOAP message is set via the *SOAP-ENV:encodingStyle* attribute.

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
  xmlns="http://schemas.xmlsoap.org/soap/envelope/">

    < SOAP-ENV:Header>
     ...
    </ SOAP-ENV:Header>

    < SOAP-ENV:Body>
        <SOAP-ENV:Fault>
<faultcode xsi:type="xsd:string">SOAP-ENV:Client</faultcode>

   <faultstring xsi:type="xsd:string">
            Failed to locate method (ValidateCreditCard) in class
(examplesCreditCard) at /usr/local/ActivePerl-
5.6/lib/site_perl/5.6.0/SOAP/Lite.pm line 1555.
     </faultstring>

       </SOAP-ENV:Fault>
</ SOAP-ENV:Body>
   ...
</ SOAP-ENV: Envelope>
```

WSDL

WSDL stands for Web Services Description Language. It is the standard format for describing a web service. WSDL was developed jointly by Microsoft and IBM.

- WSDL is an XML-based protocol for information exchange in decentralized and distributed environments.
- WSDL definitions describe how to access a web service and what operations it will perform.
- WSDL is a language for describing how to interface with XML-based services.
- WSDL is an integral part of Universal Description, Discovery, and Integration (UDDI), an XML-based worldwide business registry.
- WSDL is the language that UDDI uses.
- WSDL is pronounced as 'wiz-dull' and spelled out as 'W-S-D-L'.

WSDL

WSDL document uses the following elements in the definition of network services:

- **Definition** : It is the root element of all WSDL documents. It defines the name of the web service, declares multiple namespaces used throughout the remainder of the document, and contains all the service elements described here.
- **Types**– a container for data type definitions using some type system (such as XSD).
- **Message**– an abstract, typed definition of the data being communicated.
- **Operation**– an abstract description of an action supported by the service.
- **Port Type**–an abstract set of operations supported by one or more endpoints.
- **Binding**– a concrete protocol and data format specification for a particular port type.
- **Port**– a single endpoint defined as a combination of a binding and a network address.
- **Service**– a collection of related endpoints.

WSDL

The **<definitions>** element must be the root element of all WSDL documents. It defines the name of the web service. It is the container for all the other elements.

Example:

```
<definitions name="HelloService"
targetNamespace="http://www.examples.com/wsdl/HelloService.wsdl"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://www.examples.com/wsdl/HelloService.wsdl"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
................................................
</definitions>
```

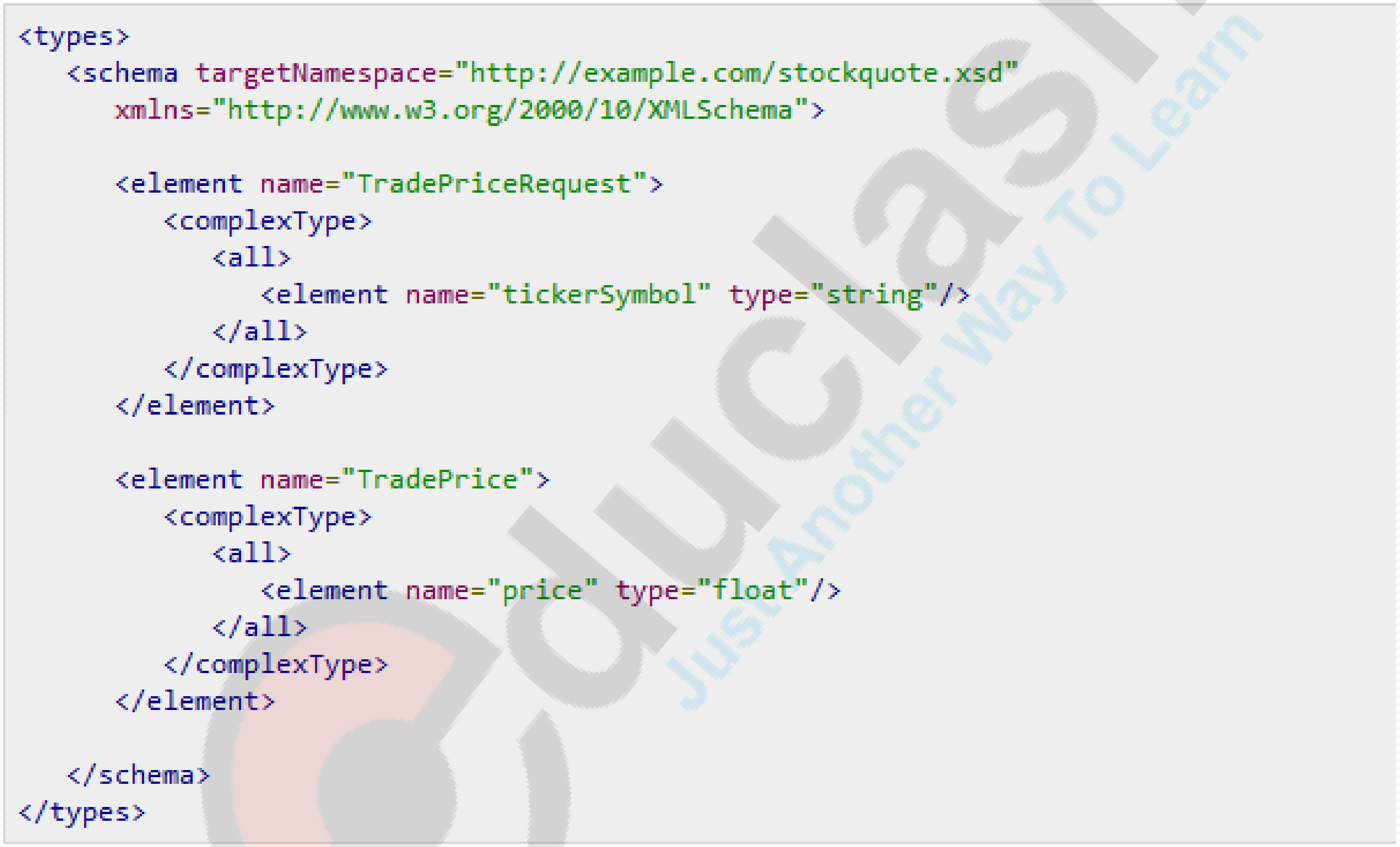
WSDL

A web service needs to define its inputs and outputs and how they are mapped into and out of the services. WSDL **<types>** element takes care of defining the data types that are used by the web service. Types are XML documents, or document parts.

- The types element describes all the data types used between the client and the server.
- WSDL is not tied exclusively to a specific typing system.
- WSDL uses the W3C XML Schema specification as its default choice to define data types.
- If the service uses only XML Schema built-in simple types, such as strings and integers, then *types* element is not required.
- WSDL allows the types to be defined in separate elements so that the types are reusable with multiple web services.

WSDL

Example:-



```
<types>
   <schema targetNamespace="http://example.com/stockquote.xsd"
      xmlns="http://www.w3.org/2000/10/XMLSchema">

      <element name="TradePriceRequest">
         <complexType>
            <all>
               <element name="tickerSymbol" type="string"/>
            </all>
         </complexType>
      </element>

      <element name="TradePrice">
         <complexType>
            <all>
               <element name="price" type="float"/>
            </all>
         </complexType>
      </element>

   </schema>
</types>
```

WSDL

- The <**message**> element describes the data being exchanged between the web service providers and the consumers.
- Each Web Service has two messages: input and output.
- The input describes the parameters for the web service and the output describes the return data from the web service.
- Each message contains zero or more <**part**> parameters, one for each parameter of the web service function.
- Each <**part**> parameter associates with a concrete type defined in the <**type**> container element.

WSDL

```
<message name="SayHelloRequest">
   <part name="firstName" type="xsd:string"/>
</message>

<message name="SayHelloResponse">
   <part name="greeting" type="xsd:string"/>
</message>
```

WSDL

The **<portType>** element combines multiple message elements to form a complete one-way or round-trip operation.

WSDL supports four basic patterns of operation:

- **One-way**

The service receives a message. The operation therefore has a single *input* element.

- **Request-response**

The service receives a message and sends a response. The operation therefore has one *input* element, followed by one *output* element. To encapsulate errors, an optional *fault* element can also be specified.

- **Solicit-response**

The service sends a message and receives a response. The operation therefore has one *output* element, followed by one *input* element. To encapsulate errors, an optional *fault* element can also be specified.

- **Notification**

The service sends a message. The operation therefore has a single *output*element.

WSDL

The **<binding>** element provides specific details on how a *portType* operation will actually be transmitted over the wire.

- The bindings can be made available via multiple transports including HTTP GET, HTTP POST, or SOAP.
- The bindings provide concrete information on what protocol is being used to transfer *portType* operations.
- The bindings provide information where the service is located.
- For SOAP protocol, the binding is **<soap:binding>**, and the transport is SOAP messages on top of HTTP protocol.
- You can specify multiple bindings for a single *portType*.

WSDL

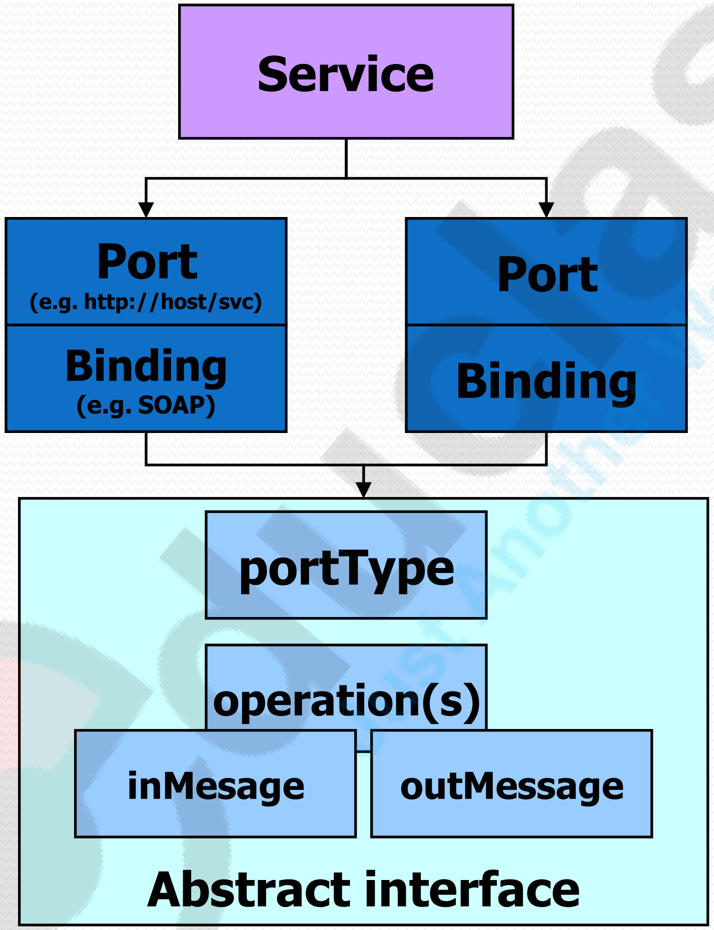
The binding element has two attributes : *name* and *type* attribute.

<binding name="Hello_Binding" type="tns:Hello_PortType">

WSDL

A **<port>** element defines an individual endpoint by specifying a single address for a binding.

- The port element has two attributes: *name* and *binding* .
- The name attribute provides a unique name among all ports defined within the enclosing WSDL document.
- The binding attribute refers to the binding using the linking rules defined by WSDL.
- Binding extensibility elements are used to specify the address information for the port.
- A port MUST NOT specify more than one address.
- A port MUST NOT specify any binding information other than address information.


Service
Port
(e.g. http://host/svc)
Binding
(e.g. SOAP)
Port
Binding
portType
operation(s)
inMesage
outMessage
Abstract interface

WSDL

The **<service>** element defines the ports supported by the web service. For each of the supported protocols, there is one port element. The service element is a collection of ports.

Web service clients can learn the following from the service element:

- where to access the service,
- through which port to access the web service, and
- how the communication messages are defined.

The service element includes a documentation element to provide human-readable documentation.

UDDI

UDDI is an XML-based standard for describing, publishing, and finding web services.

- UDDI stands for **Universal Description, Discovery, and Integration.**
- UDDI is a specification for a distributed registry of web services.
- UDDI is a platform-independent, open framework.
- UDDI can communicate via SOAP, CORBA, Java RMI Protocol.
- UDDI uses Web Service Definition Language(WSDL) to describe interfaces to web services.
- UDDI is seen with SOAP and WSDL as one of the three foundation standards of web services.
- UDDI is an open industry initiative, enabling businesses to discover each other and define how they interact over the Internet.

UDDI

UDDI has two sections:

- A registry of all web service's metadata, including a pointer to the WSDL description of a service.
- A set of WSDL port type definitions for manipulating and searching that registry.

UDDI

The UDDI Business Registry system consists of three directories:

- UDDI white pages: basic information such as a company name, address, and phone numbers, as well as other standard business identifiers like Dun & Bradstreet and tax numbers.
- UDDI yellow pages: detailed business data, organized by relevant business classifications. The UDDI version of the yellow pages classifies businesses according to the newer NAICS (North American Industry Classification System) codes, as opposed to the SIC (Standard Industrial Classification) codes.
- UDDI green pages: information about a company's key business processes, such as operating platform, supported programs, purchasing methods, shipping and billing requirements, and other higher-level business protocols.

UDDI

The UDDI technical architecture consists of three parts:

- **UDDI Data Model**

UDDI Data Model is an XML Schema for describing businesses and web services.

- **UDDI API Specification**

It is a specification of API for searching and publishing UDDI data.

- **UDDI Cloud Services**

These are operator sites that provide implementations of the UDDI specification and synchronize all data on a scheduled basis.