Web Services

K.L. Lin
Agenda

- Introduction to Web Services
- Introduction to XML
- Introduction to SOAP
- Introduction to WSDL
- Introduction UDDI
- Demo
Evolution of e-Business

Access | Publish | Transact | Integrate Internally | Integrate Externally | Adapt Dynamically

Get on the Net

Buying not browsing
Working not surfing

On Demand

The ability to respond with speed to any customer demand, market opportunity or competitive threat.
Yes, we have it in stock

Your order# is 23625336

We can deliver it next Friday

Product Information
Product Availability
Inventory

Create Order
Order Status
Orders

Schedule Delivery
Delivery Status
Deliveries

Simple Service Scenario, Order Processing
We Need Rapid and Dynamic Interoperation

Need to interoperate within our enterprise
We Need Rapid and Dynamic Interoperation

Need to interoperate within our enterprise

And interoperate between enterprise
Solution

Web Services
The Based Technologies of Web

- **TCP/IP**
  - The universal networking protocol. Everything from pagers to mobile phones to laptops to mainframes can communicate through TCP/IP.

- **HTML**
  - The universal user interface. You can use HTML tags to render data on just about any device you can think of.

- **Java**
  - The language for universal code. Write your code once, and it runs flawlessly everywhere with great performance.

- **XML**
  - The language for universal data. XML documents make it easy to move structured data across the Web.

- **What do these technologies have in common? They're all open, cross-platform standards.** It's openness that makes the Web possible; that openness is also the foundation for Web services.
Important Trends of Web

- Content is becoming more dynamic.
- Bandwidth is getting cheaper.
- Storage is getting cheaper.
- Pervasive computing is becoming more important.
Where do Web services fit in?

- **Content is becoming more dynamic**
  - A Web service has to be able to combine content from many different sources.

- **Bandwidth is getting cheaper**
  - As bandwidth continues to grow, Web services must adapt to new content types.

- **Storage is getting cheaper.**
  - A Web service must be able to deal with massive amounts of data intelligently.

- **Pervasive computing is becoming more important.**
  - Web services have to serve all sorts of devices, platforms, and browser types, delivering content over a wide variety of connection types.
What are Web Services?

- Web Services:
  - are accessible over the Web using well-known protocols
  - are described using XML
  - interact via XML
  - locatable via a registry
What are Web Services?

- Web Services:
  - are accessible over the Web using well-known protocols
    - HTTP
  - are described using XML
    - WSDL
  - interact via XML
    - SOAP
  - locatable via a registry
    - UDDI
Why Web Services?

Using Web Services when you need to:

- gain interoperability among distributed application
- gain access to applications through firewalls
- intermix different development environments and languages
- improve enterprise agility & generate new revenue streams
Examples of Web Services

Google

amazon.com
Roles of Web Services

- **Service providers**
  - Provide services, and maintain a registry that makes those services available.

- **Service brokers**
  - Clearinghouses for services. Service brokers act as matchmakers between service providers and service requestors.

- **Service requestors**
  - Work with service brokers to discover Web services, then invoke those services to create applications.
Operations of Web Services

- **Publish/Unpublish**
  - Publishing and unpublishing involves advertising services to a registry (publishing) or removing those entries (unpublishing). The service provider contacts the service broker to publish or unpublish a service.

- **Find**
  - The find operation is performed by service requestors and service brokers together. The service requestors describe the kinds of services they're looking for, and the service brokers deliver the results that best match the request.

- **Bind**
  - The bind operation takes place between the service requestor and the service provider. The two parties negotiate as appropriate so the requestor can access and invoke services of the provider.
Service Oriented Architecture

Service Requestors → Internet

Service Brokers

Service Providers → Internet

Legacy System

Find

Bind

Publish
Web Services Technology Stack

- Service Flow (BPEL)
- Service Discovery & Publication (UDDI)
- Service Description (WSDL)
- Service Invocation & Messaging (SOAP)
- XML (DTD, Namespace, Schema)
- Network (HTTP, SMTP, FTP)
- Security (WS-Sec)
- Quality of Service
- Management
Components of Web Services

- **XML (eXtensible Markup Language)**
  - A text-based mark-up language for structuring data content into a self-describing wrapper.
  - It was designed to create documents and data records that are fully portable and platform independent.

- **WSDL (Web Service Description Language)**
  - An XML-based description of what a Web service can do and how to invoke it.
  - It represents a contract between the service requestors and the service providers.
Components of Web Services

- **SOAP (Simple Object Access Protocol)**
  - An XML-based communication protocol and encoding format for inter-application communication.
  - It was designed to information exchange.

- **UDDI (Universal Description Discovery and Integration)**
  - An XML-based framework that allows sharing, finding, and registering of global business information.
  - Enables businesses to quickly, easily, and dynamically publish and locate web services.
# Web Services Standards

<table>
<thead>
<tr>
<th>Standard</th>
<th>WSDL</th>
<th>SOAP</th>
<th>UDDI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maintain by</td>
<td>W3C</td>
<td>W3C</td>
<td>OASIS</td>
</tr>
<tr>
<td>Recent status</td>
<td>1.2</td>
<td>1.2</td>
<td>3.0</td>
</tr>
</tbody>
</table>
Web Services Architecture

**White Pages:**
Service provider information

**Yellow Pages:**
Business categories

**Green Pages:**
How to conduct the binding

- Business identifier
- Classification
- Service description
- URL
- Binding information
- Interface
- Properties
...

**WSDL**
- Types
- Message
- Operation
- Port type
- Binding
- Port
- Service

**UDDI/WSDL**

**SOAP**
Feature of Web Services

- Services as components
- Platform-free
- Open standard
- Dynamic Integration
- Interoperability
- Incremental Deployment
Definition of Web Services

- A Web service is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols.
Introduction to XML

- Markup Language
- HTML
- XML
- XML Component
- HTML vs. XML
Markup Language

- What is markup language?
  - A 'metalanguage' — a language for describing other languages.
HTML

What is HTML?

- HTML (HyperText Markup Language) is a metalanguage to specify the structure of documents for retrieval across the Internet using browser programs of the Web.

```html
<html>
  <head>
    <title>BBC News</title>
  </head>
  <body>
    <h2>Oldest flower’ …
    </h2>
    …..
  </body>
</html>
```
XML

- Overview of XML
- Overview of DTD
Overview of XML

- What is XML?
- What is an XML Document?
- XML Example – Book
- What is well-formed XML?
- What is a valid XML document?
What is XML?

- XML (Extensible Markup Language) is designed to improve the functionality of the Web by providing more flexible and adaptable information identification.
- It is called extensible because it is not a fixed format like HTML.
- It can be used to store any kind of structured information, and to enclose or encapsulate information in order to pass it between different computing systems which would otherwise be unable to communicate.
What is an XML Document?

An XML document primarily consists of a strictly nested hierarchy of elements with a single root. Elements can contain character data, child elements, or a mixture of both. In addition, they can have attributes. Child character data and child elements are strictly ordered; attributes are not.
<?xml version="1.0" ?>
<Book Author="Anonymous">
  <Title>Sample Book</Title>
  <Chapter id="1">
    This is chapter 1. It is not very long or interesting.
  </Chapter>
  <Chapter id="2">
    This is chapter 2. Although it is longer than chapter 1, it is not any more interesting.
  </Chapter>
</Book>
What is well-formed XML?

- **Syntax of XML:**
  - All XML elements must have a closing tag
  - XML tags are case sensitive
  - All XML elements must be properly nested
  - All XML documents must have a root tag
  - Attribute values must always be quoted

- A "Well Formed" XML document is a document that conforms to the XML syntax rules.
What is a valid XML document?

A "Valid" XML document is a "Well Formed" XML document which conforms to the rules of a Document Type Definition (DTD).
Overview of DTD

- What is DTD?
- The Declaration of DTD
- DTD Example – Book
- DTD Element
- Attributes of a Element
- Namespace
What is DTD?

- A DTD is a set of rules that defines:
  - What tags appear in a XML document
  - What attributes the tags may have
  - What a relationship the tags have with each other

- When an XML document is processed, it is compared within the DTD to be sure it is structured correctly and all tags are used in the proper manner. This comparison process is called validation and it is performed by a tool called parser.
The Declaration of DTD

- A DTD can be declared inline in your XML document, or as an external reference.
  - Internal DOCTYPE declaration
    ```xml
    <!DOCTYPE root-element [element-declarations]>
    ```
  - External DOCTYPE declaration
    ```xml
    <!DOCTYPE root-element SYSTEM "filename">
    ```
<!DOCTYPE Book [
  <!ELEMENT Book (Title, Chapter+)> 
  <!ATTLIST Book Author CDATA #REQUIRED> 
  <!ELEMENT Title (#PCDATA)> 
  <!ELEMENT Chapter (#PCDATA)> 
  <!ATTLIST Chapter id ID #REQUIRED> 
]>
The purpose of a DTD is to define the legal building blocks of an XML document. It defines the document structure with a list of legal elements.
DTD Element Overview

- Syntax of element declaration
- Content of element
- Operators of element
Syntax of Element Declaration

<!ELEMENT element-name category> or
<!ELEMENT element-name (element-content)>
Content of Element

- Text content
- Subelement content
- Empty content
- Any content
- Mixed content
Elements with only character data are declared with \texttt{#PCDATA} inside parentheses

\begin{verbatim}
<!ELEMENT element-name (#PCDATA)>

example:
<!ELEMENT from (#PCDATA)>

XML example:
<from>Taipei</from>
\end{verbatim}
Subelement Content

- Elements with one or more children are defined with the name of the children elements inside parentheses:
Subelement Content

<!ELEMENT element-name (child-element-name)> or
<!ELEMENT element-name (child-element-name, child-element-name, .....)> 

example:

<!ELEMENT note (to, from, header, body)> 
<!ELEMENT to (#PCDATA)> 
<!ELEMENT from (#PCDATA)> 
<!ELEMENT heading (#PCDATA)> 
<!ELEMENT body (#PCDATA)> 

XML example:

<note>
<to>Kaohsiung</to>
<from>Taipei</from>
<header>Birthday Card</header>
<body>Happy Birthday</body>
</note>
Empty Content

- Empty elements are declared with the category keyword **EMPTY**

```xml
<!ELEMENT element-name EMPTY>
```

**example:**

```xml
<!ELEMENT br EMPTY>
```

**XML example:**

```xml
<br/>
```
Any Content

- Elements declared with the category keyword **ANY** can contain any combination of parsable data

```xml
<!ELEMENT element-name ANY>

example:

<!ELEMENT note ANY>
```
Mixed Content

Content of element can containing a mixture of character data and elements in any order.

example:

```xml
<!ELEMENT letter (#PCDATA | name | orderid | shipdate)*>  

XML example:

<letter>
Dear Mr.<name>John Smith</name>.  
Your order <orderid>1032</orderid>  
will be shipped on <shipdate>2001-07-13</shipdate>.  
</letter>
```
Operators of Element

- One-or-more (+) operator
- Zero-or-more (*) operator
- Optional (?) operator
- Either/or (|) operator
One-or-more (+) operators

Declaring **minimum one occurrence** of the same element

```xml
<!ELEMENT element-name (child-name+)>  
example:
<!ELEMENT book (chapter+)>  
XML example:
<book>
  <chapter>chapter 1</chapter>
  <chapter>chapter 2</chapter>
</book>
```
Zero-or-more (*) operator

- Declaring zero or more occurrences of the same element

```xml
<!ELEMENT element-name (child-name*)>

example:
<!ELEMENT book (chapter*)>

XML example:
<book>
  </book>
```
Optional (?) operator

- Declaring zero or one occurrences of the same element

```xml
<!ELEMENT element-name (child-name?)>
```

example:

```xml
<!ELEMENT book (chapter?)>
```

XML example:

```xml
<book>
  <chapter>chapter 1</chapter>
</book>
```
Either/or (|) operator

- Content of element can be either a sequence or choice of subelements and/or subgroups.

```xml
<!ELEMENT element-name (child1-name | child2-name | …)>  

example:

<!ELEMENT book (name , ( author | translator ) )>  

XML example:

<book>
  <name>C++</name>
  <translator>Bill</translator>
</book>
```
PCDATA

- "PCDATA" in the declarations is short for "Parsed Character DATA".
- In the XML document following the element tag is parsed looking for more markup tags.

<!ELEMENT A (#PCDATA)>
Attributes of a Element

- Attributes provide **extra information about elements**.
- Attributes are always placed inside the starting tag of an element. Attributes always come in **name/value pairs**.

```xml
<!ATTLIST element-name attribute-name attribute-type default-value>
```

Example:

```xml
<!ELEMENT name #PCDATA>
<!ATTLIST name gender ("male" | "female") #REQUIRED>
```

XML example:

```
<name gender="male">Tom</name>
```
### Possible value of Attribute Type

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDATA</td>
<td>The value is character data</td>
</tr>
<tr>
<td>(en1</td>
<td>en2</td>
</tr>
<tr>
<td>ID</td>
<td>The value is a unique id</td>
</tr>
<tr>
<td>IDREF</td>
<td>The value is the id of another element</td>
</tr>
<tr>
<td>IDREFS</td>
<td>The value is a list of other ids</td>
</tr>
<tr>
<td>NMTOKEN</td>
<td>The value is a valid XML name</td>
</tr>
<tr>
<td>NMTOKENS</td>
<td>The value is a list of valid XML names</td>
</tr>
<tr>
<td>ENTITY</td>
<td>The value is an entity</td>
</tr>
<tr>
<td>ENTITIES</td>
<td>The value is a list of entities</td>
</tr>
<tr>
<td>NOTATION</td>
<td>The value is a name of a notation</td>
</tr>
<tr>
<td>xml:</td>
<td>The value is a predefined xml value</td>
</tr>
</tbody>
</table>
Possible value of Default Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>The default value of the attribute</td>
</tr>
<tr>
<td>#REQUIRED</td>
<td>The attribute value must be included in the element</td>
</tr>
<tr>
<td>#IMPLIED</td>
<td>The attribute does not have to be included</td>
</tr>
<tr>
<td>#FIXED value</td>
<td>The attribute value is fixed</td>
</tr>
</tbody>
</table>
XML Schema

- Information modeling language for XML developed by W3C
- Full support for datatypes
  - Built in types (integer, boolean, etc.)
  - Custom types (telephone numbers, etc.)
Schema vs. Schemas vs. schema

- **Schema**
  - *Refers to the W3C XML Schema*

- **Schemas**
  - *Refers to one or more XML vocabularies*
    - e.g. xCBL, CXML, BizTalk, IFX, etc. etc.

- **schema (Lowercase)**
  - *Refers to an information model*
  - *Can be a DTD or Schema*
  - *May also be a relational database schema*
Comparing Element Declarations

- Differences in data typing and syntax

<table>
<thead>
<tr>
<th>DTD</th>
<th>&lt;!ELEMENT Age (#PCDATA)&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML Schema</td>
<td>&lt;element name=&quot;Age&quot; type=&quot;integer&quot;/&gt;</td>
</tr>
<tr>
<td>XML</td>
<td>&lt;Age&gt;18&lt;/Age&gt;</td>
</tr>
</tbody>
</table>
"a is the parent of b. x and y are children of b."  x and y are always interpreted as String, regardless of how they are used.
XML Schema

Information Modeling Syntax in XML

<xsl:element name="x" type="boolean" />
<xsl:attribute name="z" type="integer" />

x and y now have enforceable data types
Schema Validation

- Schemas can be used to validate a document in two ways:
  - Content Model Validation
    - Checks order and nesting of elements (similar to DTD validation)
  - DataType Validation
    - Checks the element content for valid type and range

- Example:
  - Month element is an integer between 1 and 12
    - `<month>5</month>`  VALID!!
    - `<month>15</month>` INVALID!!
Comparison with DTD

- XML Schema uses XML syntax
  - DTD use their own unique syntax (SGML)
- XML Schema is verbose
  - DTD are concise
- XML Schema can be parsed and manipulated programmatically like any other XML document
  - DTD cannot
- XML Schema enforce Data Typing DTD cannot
  - DTD see everything as Strings
- XML Schema supports namespace integration
Multiple Levels of Checking

- **BookCatalogue.xml**
- **BookCatalogue1.xsd**
- **xml-schema.dtd**

**Questions:**
- Does the xml document conform to the rules laid out in the xml-schema?
- Is the xml-schema a valid xml document, i.e., does it conform to the rules laid out in the xml-schema DTD?
End of DTD?

No!

- DTD have:
  - Widespread use and support
  - Many legacy applications, documents
  - Too much time & money invested
  - Experienced programmers/consultants
  - There is a DTD to define XML Schema
XML Namespace

- Since element names in XML are not fixed, very often a name conflict will occur when two different documents use the same names describing two different types of elements.

- XML Namespaces provide a simple method for qualifying element and attribute names used in XML documents by associating them with namespaces identified by URI references.
XML Namespaces

- Two-part naming convention:
  - The local name
  - The URI of the XML namespace

```xml
<x xmlns:foo="http://www.foo.org/">
  <edi:price units='Euro'>32.18</edi:price>
</x>
```
HTML vs. XML

**Does XML replace HTML?**

- No. XML itself does not replace HTML: instead, it provides an alternative which allows you to define your own set of markup elements.

```html
<html xmlns:myHtml='http://www.mysite.com'>
  <myHtml:head>
    <myHtml:title>Hello</myHtml:title>
  </myHtml:head>
  <myHtml:body>
    <myHtml:p>Hello World</myHtml:p>
  </myHtml:body>
</html>
```
SOAP

- Concept of SOAP
- Benefit and Issues of SOAP
- Message Structure of SOAP
- Example
Distribute Computing Environment

- **Transparency**
  - 對使用者而言，將永遠是如同單一系統，永遠不用花心思去了解它的整個結構如何複雜，如Printer在哪一個系統，哪一個地方，Database Server，Computing Server等又是在哪裡。這一層透明化將由命名服務(Naming Service或稱之為Directory Service)來處理。

- **Interoperability**
  - 將提供一個在異種（不同）的硬體環境，異種的操作環境和異種的軟體環境裡，能互通。也就是一般稱的heterogeneous或multi-vendor environments。

- **Scalability**
  - 系統除了維護外，總會不斷有新的功能、資料加入。在DCE，延展性是絕對須要，且必須要能很順利的成長，並有廣大的成長空間。
Distribute Computing Environment

- **Portability**
  - 由于DCE一定是主/从(Client/Server)的架构。所以使用者在DCE可以很容易地将Server上的资料，搬至Client上，而Client也可以就是一个可携式的个人电脑，随著使用者因工作上的需要而旅行，而远离它的Server一段时间。一段时间后Client与Server再连在一起，而彼此仍能达到一致性，共享资源。

- **Security**

- **Reliability, Availability and High Performance**
  - 也是DCE的重要一环。使用者永远是要求高稳定性、高效率。不论他（她）们在何时、何地，使用何种样电脑。
History of Distributed Computing

- EMAIL
- EDI
- Minicomputer
- RPC
- CORBA, DCOM
- The Web
- XML AND Related
- eSPEAK, XML-RPC
- SOAP & WSDL
- WS Stack

Timeline:

- 1970
- 1980
- 1990
- 1997
- 1998
- 1999
- 2000
- 2001
- 2002
Motivation of SOAP

- Many object models out there
  - CORBA (Common Object Request Broker Architecture)
    - Support by OMG
    - Default port number 135
  - DCOM (Distributed Common Object Model)
    - Support by Microsoft
  - RMI (Remote Method Invocation)
    - Support by SUN
    - Default port number 1099
Motivation of SOAP

- They use different "on the wire" protocol
- They are complex in varying degree
- It is hard to make different object model to inter-operate
Comparison of SOAP with other Distributed Architectures

<table>
<thead>
<tr>
<th>Protocol name</th>
<th>CORBA</th>
<th>DCOM</th>
<th>Java-RMI</th>
<th>SOAP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scalability</strong></td>
<td>Corba uses stateful programming model which is not as scalable.</td>
<td>Least scalable. Clients ping the server at regular intervals to ascertain that it is still available. This pinging process limits scalability when large # of connections are involved.</td>
<td>Relatively scalable. Uses RMI Registry which could limit scalability if it is located on one server.</td>
<td>Most scalable of the four.</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td>Once an object reference is obtained, CORBA permits direct client-server communication. Hence subsequent communication is very fast.</td>
<td>Requires several round-trips to activate and use the remote object. Once object’s reference is obtained, direct object access without DCOM can take place from client.</td>
<td>Good performance. Works for Java language only and hence is fine-tuned for it.</td>
<td>Currently low. Overhead of extracting SOAP envelope, parsing XML, creating appropriate objects and converting parameters.</td>
</tr>
</tbody>
</table>
## Comparison of SOAP with other Distributed Architectures

<table>
<thead>
<tr>
<th>State Mgmt.</th>
<th>CORBA</th>
<th>DCOM</th>
<th>Java-RMI</th>
<th>SOAP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Connection - oriented</td>
<td>Provides location transparency.</td>
<td>Very flexible. Provides both stateful and</td>
<td>Not addressed by SOAP. If HTTP is the</td>
</tr>
<tr>
<td></td>
<td>and stateful.</td>
<td>Is stateful.</td>
<td>stateless sub-protocols.</td>
<td>protocol used, it is stateless.</td>
</tr>
<tr>
<td>Garbage collection</td>
<td>CORBA does not address</td>
<td>Provides automatic garbage</td>
<td>Excellent garbage collection</td>
<td>SOAP does not address garbage collection.</td>
</tr>
<tr>
<td></td>
<td>distributed memory</td>
<td>collection using the pinging</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>management.</td>
<td>mechanism discussed earlier.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security</td>
<td>No intrinsic support for</td>
<td>Very security-oriented.</td>
<td>Since Java RMI works with java programming</td>
<td>Since SOAP is a wire protocol, it does</td>
</tr>
<tr>
<td></td>
<td>authentication,</td>
<td>Provides support for authentication,</td>
<td>language it inherits the security built into</td>
<td>not address security. Security is</td>
</tr>
<tr>
<td></td>
<td>authorization or identity.</td>
<td>authorization or identity. User</td>
<td>Java. Use of RMI Security Manager can enable</td>
<td>determined by the transport protocol</td>
</tr>
<tr>
<td></td>
<td></td>
<td>can set appropriate level of</td>
<td>dynamic class loading thus providing</td>
<td>that it uses. For example, HTTPS using</td>
</tr>
<tr>
<td></td>
<td></td>
<td>security.</td>
<td>additional security.</td>
<td>secured socket layer (SSL) when HTTP is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>the transport protocol.</td>
</tr>
</tbody>
</table>
What is SOAP?

- Wire protocol similar to
  - IIOP for CORBA
  - JRMP for RMI
  - DRPC for DCOM

- XML is used for data encoding
  - SOAP uses "text" based protocol as opposed to "binary" protocol

- Support XML-based RPC
- Runs over HTTP, SMTP
- Support by IBM, Microsoft, W3C
History of SOAP

- 1998 年，UserLand 公司的執行總裁 Dave Winer 提出透過 XML 讓 RPC 的通訊方式透過 HTTP 協定在網際網路上執行
  - XML-RPC
- 經由微軟公司加以改良，提出了實際可行的 Simple Object Access Protocol (SOAP) 通訊協定
- 2000 年由 IBM、Microsoft、UserLand 和 DevelopMentor 共同提交給 W3C
SOAP Goals

- Enterprise Application Integration
- Supply Chain Automation
Message Structure of SOAP

- Request and Response messages
  - Request invokes a method on a remote object
  - Response returns result of running the method

- SOAP defines an "envelope"
  - "envelope" wraps the message itself
  - message is a different vocabulary
  - namespace prefix is used to distinguish the two parts message
Structure of SOAP Message

- SOAP Envelope
  - SOAP Header
    - Extended Information
  - SOAP Body
    - Method Element
    - Embedded Elements
    - Other Independent Elements
Benefit of SOAP

- Simple
- Light-weight
- Firewall-friendly
- Vendor-neutral
- Enables inter-operability
- Hides the service implementation from requester
standard XML request/response messages
Example: Remote Procedure

```java
public interface Hello {
    public String sayHelloTo(String name);
}
```
Simplified SOAP Request

```xml
<?xml version="1.0"?>
<Hello>
  <sayHelloTo>
    <name>John</name>
  </sayHelloTo>
</Hello>
```
<?xml version="1.0"?>
<Hello>
    <sayHelloToResponse>
        <message>
            Hello John, How are you?
        </message>
    </sayHelloToResponse>
</Hello>
SOAP Response

```xml
<SOAP-ENV:Envelope xmlns:SOAPENV=
http://schemas.xmlsoap.org/soap/envelope/
xmlns:xsd=http://www.w3.org/1999/XMLSchema>
  <SOAP-ENV:Header> </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <ns1:sayHelloToResponse xmlns:ns1="Hello"
      SOAP-ENV:encodingStyle=
        "http://schemas.xmlsoap.org/soap/encoding/">
      <return type="xsd:string">
        Hello John, How are you doing?
      </return>
    </ns1:sayHelloToResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
SOAP consists of

- **Envelope**
  - define a framework for describing what is in a message and how to process it

- **Encoding rules**
  - expressing instances of application-defined datatypes

- **RPC representation**
  - representing RPCs and responses
Envelope

- Header
  - who should deal with a feature (SOAP actor)
  - where it is optional or mandatory (mustUnderstand)

- Body
  - container for mandatory information
  - include following information
    - RPC calls
    - Fault element
Encoding rules

- based on a simple type system that is a generalization of the common features
- operates on two levels
  - given a schema consistent with the type system described, a schema for an XML grammar maybe constructor
  - given a schema and a particular graph of value, an XML instance may be constructed
- “http://schemas.xmlsoap.org/soap/encoding”
Encoding rules

The value of type is determined by

- xsi:type attribute
- reference to a schema (notation is described by XML Schema)

Simple value

- one without named parts
- a type that is listed XML Schema such as strings, integers, enumerated values, arrays of bytes
Encoding rules

- **Compound value**
  - aggregation of relations to other values
  - encoded as a sequence of elements
  - array: ordinal position servers as the distinction
  - struct: element name is the distinction
RPC representation

- To make a method call, following is needed
  - The URI of the target object
  - A method name
  - An optional method signature
  - The parameters to the method
  - Optional header data

- Method calls and responses are both carried in the SOAP Body element
SOAP/HTTP Request and Response

1. deployment
   - deployment process
   - deployment file

2. request
   - check SOAPAction Field
   - rpcrouter servlet

3. result
   - reading xml & call method
   - encoding to xml

Client
- encoding to xml
- decoding from xml
WSDL

- Describe XML-based network services regardless of communication protocol and request structure
- XML equivalent of a resume
  - what a web service can do
  - where it resides
  - how to invoke
- WSDL is the convergence of IBM's NASSL and Microsoft's SDL
  - Some Web Services docs may refer to old names
WSDL Structure

- `<definitions> element : contains the definition of one or more services

```xml
<definitions name="Weather_Service">
  <message> ... 
  <portType> ... 
  <binding> ... How the operations are invoked 
  <service> ... Where the service is located 
</definitions>
```
WSDL Structure

- What operations the service provides

```xml
<message name = "IngetTempRequest">
   <part name="meth1_inType1" type="xsd:string"/>
</message>
<message name = "IngetTempResponse">
   <part name="meth1_outType1" type="xsd:string"/>
</message>
<portType name="Weather_Service">
   <operation name="getTemp">
      <input message="IngetTempRequest"/>
      <output message="IngetTempResponse"/>
   </operation>
</portType>
```
WSDL Structure

- **How the operations are invoked**

```xml
<binding name="Weather_ServiceBinding"
type="Weather_Service">
  <soap:binding style="rpc" />
  <operation name="getTemp">
    <soap:operation soapAction="urn:weather-service"/>
    <input>
      <soap:body encodingStyle="..."
        namespace="urn:weather-service"
        use="encoded"/>
    </input>
    <output>
      <soap:body encodingStyle="..."
        namespace="urn:weather-service"
        use="encoded"/>
    </output>
  </operation>
</binding>
```
WSDL Structure

- *Where* the service is located

```xml
<service name="Weather_Service">
    <port binding="Weather_ServiceBinding"
         name="Weather_ServicePort">
        <soap:address
            location="http://localhost:8080/soap/servlet/rcpro
ter"/>
    </port>
</service>
```
UDDI

- Universal Description, Discovery, and Integration
- A project to speed interoperability and adoption for web services
  - Standards-based specifications for service description and discovery
  - Shared operation of a web-based business registry
  - Partnership among industry and business leaders - more than 70 companies have signed up so far
UDDI.org

- Discover businesses worldwide that offer the exact products and services that you need.
- Register the products and services of your own business for others to discover. Or both.
- Technology and business champions are leading the development and deployment of an open, Internet-based Universal Description, Discovery, and Integration (UDDI) specification.
- UDDI is the building block that will enable businesses to quickly, easily and dynamically find and transact business with one another using their preferred applications.
Problems UDDI Solves

Broader B2B

A mid-sized manufacturer needs to create 400 online relationships with customers, each with their own set of standard and protocols

Smarter Search

A flower shop in Australia wants to be “plugged in” to every marketplace in the world, but doesn’t know how

Easier Aggregation

A B2B marketplace cannot get catalog data for relevant suppliers in its industry, along with connections to shippers, insurers, etc.
Registry Data

- White pages: information such as the name, address, and telephone number of a given business
- Yellow pages: information that categorizes business. Based on existing standards
- Green pages: technical information about the web services provided by a given business
Demo – Hello Web Services

Development environment

- **Server**: To develop a ASP.NET Web Services project
  - Win 2K Server
  - IIS (Internet Information Services)
    - Web Server
  - VS.Net (Visual Studio .Net)

- **Client**: To develop a Windows Application
  - VS.Net
Imports System.Web.Services

<WebService(Namespace:="http://tempuri.org/")>
Public Class Hello
    Web Services Designer Generated Code
    <WebMethod()> Public Function Hello(ByVal strName As String) As String
        Hello = "Hello" & strName
        End Function
    End Class
End Class
hello

hello.sln
Visual Studio Solution

修改日期: 2003/3/27 下午 09:35

大小: 929 個位元組

屬性: (一般)
管理 IIS，Internet 及 intranet 網站的 Web 伺服器。

修改日期: 2003/3/28 上午 12:06

大小: 2.24 KB

属性: (一般)
管理 IIS，Internet 及 intranet 網站的 Web 伺服器。

修改日期: 2003/3/28 上午 12:06

大小: 2.24 KB

属性: (一般)
Demo

Development environment

- Server: To develop a ASP.NET Web project
  - Win 2K Server
  - IIS (Internet Information Services)
    - Web Server
  - VS.Net (Visual Studio .Net)
- Client: To develop a Windows Application project
  - VS.Net
This tool searches for services and providers in this UDDI Services site with WSDL described services. Search by name or browse through available categorization schemes.

Service name:  
Provider name:  

Select a categorization scheme:
- unspec-org:unspec:3-1
- unspec-org:unspec
- uddi-org:iso:3166:1999
- nts-owm:ncas:1997
WS Web Service Search Categorization

Web Services

(none)
No Web References were found on this page.
Click for help on finding a Web Reference.
Hello

The following operations are supported. For a formal definition, please review the Service Description.

- Hello

This web service is using http://tempuri.org/ as its default namespace.

Recommendation: Change the default namespace before the XML Web service is made public.

Each XML Web service needs a unique namespace in order for client applications to distinguish it from other services on the Web. http://tempuri.org/ is available for XML Web services that are under development, but published XML Web services should use a more permanent namespace.

Your XML Web service should be identified by a namespace that you control. For example, you can use your company's Internet domain name as part of the namespace. Although many XML Web service namespaces look like URLs, they need not point to actual resources on the Web. (XML Web service namespaces are URIs.)
Public Class Form1
    Inherits System.Windows.Forms.Form

    Windows Form Designer generated code

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Dim objHello As New localhost.Hello()
        Label1.Text = objHello.Hello(TextBox1.Text)
    End Sub
End Class
SayHello
Visual Studio Solution

大小: 912 個位元組
属性: (一般)
Demo – Amazon Web Services

- Download Amazon Web Services 2.0
- Development environment
  - Client: To develop a Windows Application
    - VS.Net
Amazon Web Services 2.0

🌟 Web Services has launched for Amazon.co.uk! Download our new SDK now.

At Amazon, we want to see Web services work. We believe they are important to the future of the Internet. To help stimulate Web service innovation, we now offer software developers the opportunity to integrate Amazon.com and Amazon.co.uk features and content directly into other Web sites using either SOAP or XML over HTTP. Partner with the leader in e-commerce and join the Web services revolution today!

Participation Is Easy and Free

To participate in the free Amazon Web Services Program, you simply need to:

1. Download our free developer’s kit;
2. Apply for a free developer’s token;
3. Write your application.

Amazon Web Services Is Hiring!
Web Services Developer’s Kit

Our developer’s kit contains everything you need to start using Amazon Web Services. Technical documentation, sample codes—it’s all here!

Please read the Amazon.com Web Services licensing agreement before downloading the kit. This agreement will enable you to use Web Services for both Amazon.com and Amazon.co.uk.

Amazon.com (as defined below) provides Amazon.com Web Services (as defined below) to you subject to the following conditions. This is a legal agreement between you and Amazon.com. In consideration of Amazon.com allowing you to use Amazon.com Web Services and by receiving Amazon.com Properties (as defined below) or accessing Amazon.com Web Services you agree to be bound by the terms of this agreement (“Agreement”). If you do not agree, click on your browser’s back button to decline this Agreement.

Amazon.com Web Services is a platform to publish and invoke applications that perform various functions, such as retrieving information about Amazon.com products or adding a product to an Amazon.com shopping cart, wish list, or registry. Amazon.com Web Services can be accessed through two interfaces: XML over HTTP or SOAP. Both of these methods return "structured data" (product name, manufacturer, price, etc.) about products available at www.amazon.com and www.amazon.co.uk (together, the "Amazon.com Website") based on parameters such as keyword search terms and browse tree nodes. All information you receive from Amazon.com and your use of it is subject to the terms and conditions of
Three Easy Steps

1) Download the Developer's Kit.
2) Apply for a Token.
3) Write your Application!

Internet Explorer Users: Right click, Save Target As
Netscape Users: Right click, Save Link As
Macintosh Users: Click and hold, Download To Disk
Apply for a Developer's Token

You did not tick the box confirming that you read the application. Please do so and re-submit.

Applying for an Amazon Web Services developer's token is a snap—all we need is your e-mail address and a password. Please note: This token can be used for both Amazon.com and Amazon.co.uk Web Services. (If you already have an Amazon.com or Amazon.co.uk login, and you want to use it for your Web services account as well, click here).

Your e-mail: 

Enter password: 

Re-enter password: 

Please read the Amazon Web Services terms and conditions:

Amazon.com (as defined below) provides Amazon.com Web Services (as defined below) to you subject to the following conditions. This is a legal agreement between you and Amazon.com. In consideration of Amazon.com allowing you to use Amazon.com Web Services and by receiving Amazon.com Properties (as defined below) or accessing Amazon.com Web Services you agree to be bound by the terms of this agreement ("Agreement"). If you do not agree, click on your browser's back button to decline this Agreement.