# ระบบสารสนเทศผ่านเว็บเซอร์วิส

**Web Services Information System**


โดย

นายกำธร พันธุมะผล

อาจารย์ที่ปรึกษา

ดร. วรวัฒน์ ลิ้มโภคา



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ปริญญานิพนธ์ปีการศึกษา 2547

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบสารสนเทศผ่านเว็บเซอร์วิส

Web Services Information System

ผู้จัดทำ

1. นายกำธร พันธุมะผล รหัสประจำตัว 42010023

_____ อาจารย์ที่ปรึกษา

(ดร. วรวัฒน์ ลิ้มโภคา)

# การพัฒนาซอฟต์แวร์เชิงคอมโพเน็นต์โดยใช้ EJB

นายกำธร พันธุมะผล 43010023
ดร. วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษา

ปีการศึกษา 2547

บทคัดย่อ

เว็บเซอร์วิส คือ แอพพลิเคชันที่ ใช้เทคโนโลยี XML SOAP UDDI และ WSDL เว็บเซอร์วิสสามารถทำได้ตั้งแต่งานง่ายๆ เช่นเรียกข้อมูล จนถึงกระบวนการทางธุรกิจที่ซับซ้อน เช่น การทำรายการบัญชี

เนื่องจากการเติบโตของอินเตอร์เน็ต ทำให้มีการพัฒนาด้านระบบความปลอดภัยเพื่อยืนยันตัวตนของผู้ส่งความถูกต้องและการรักษาความลับของเอกสารซึ่งเทคโนโลยีที่นำมาใช้งานในการพัฒนาเว็บเซอร์วิส ในวิทยานิพนธ์ฉบับนี้คือ SSL PKI และ WS-Security โดยนำระบบความปลอดภัย SSL มาใช้งานเมื่อฝั่งผู้เรียกใช้บริการใช้บริการจากฝั่ง SERVER และนำระบบความปลอดภัย PKI และ SSL มาใช้งานเมื่อมีการเรียกใช้ เว็บเซอร์วิสจาก SERVER อื่นโดยใช้เทคโนโลยี XML-Encryption และ XML-Signature

วิทยานิพนธ์ฉบับนี้ศึกษาถึงวิธีในการพัฒนาเว็บเซอร์วิส โดยใช้จาวาเทคโนโลยีในการพัฒนา มีการใช้งานข้ามแฟลตฟอร์มร่วมกันระหว่าง .NET เฟรมเวิร์คด้วย และทำการเปรียบเทียบ เทคโนโลยีของทั้งสองค่าย

# Component-Based Software Development base on EJB

Komthorn Phuntumapon

Dr. Worawat Limpoka Advisor

ABSTRACT

Webservice is application that use XML SOAP UDDI and WSDL technology. Websevice can handle easy tasks example retrieve data from server and can handle complex tasks example Accounting.

Now internet's technology develop rapidly so security technology has develop authentication , integrity and privacy security's technology that can work with web services. example SSL PKI and WS-Security .SSL security use when client request service from server. WS-Security and PKI use when server request service from websertvice with use XML-Encryption and XML-Signature

This thesis is to study the webservice application development methodology using Java technology. Working cross platform with .NET Framework .

# กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือ และความร่วมมือจากหลายๆ ฝ่ายด้วยกัน บุคคลแรกที่ต้องกล่าวถึงเพราะเป็นส่วนสำคัญที่ทำให้ปริญญานิพนธ์นี้เสร็จลงได้ คือ ดร.วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษาปริญญานิพนธ์ ที่ให้ความเอาใจใส่ แนะนำ และความช่วยเหลือเสมอมา ซึ่งต้องขอขอบคุณเป็นอย่างมาก

ขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ คือ บิดา มารดา อันเป็นที่เคารพรักยิ่ง ซึ่งได้เลี้ยงดูผู้เขียนมาเป็นอย่างดี พร้อมทั้งให้โอกาสทางการศึกษาอย่างเต็มที่ และคอยให้กำลังใจเอาใจใส่เสมอมา ในทุกๆ ด้าน อันหาที่เปรียบมิได้ ข้าพเจ้าขอระลึกในพระคุณอันสุดประมาณ และขอกราบขอบพระคุณมา ณ ที่นี้

นายกำธร พันธุมะผล 43010023

# สารบัญ

# สารบัญตาราง

# สารบัญภาพ

# บทที่ 1
# บทนำ

**1.1 ความสำคัญและที่มา**

เว็บเซอร์วิส คือ เว็บแอพพลิเคชันยุคใหม่ สามารถติดตั้ง ค้นหา เริ่มทำงานได้ผ่านเว็บ เว็บเซอร์วิสสามารถทำได้ตั้งแต่งานง่ายๆ เช่นเรียกข้อมูล จนถึงกระบวนการทางธุรกิจที่ซับซ้อน เช่น การทำรายการบัญชี

หลายคนอาจจะถามว่าทำไมต้องเป็นเว็บเซอร์วิส เนื่องจากเว็บมีจุดเด่นในเรื่องของการให้บริการข้อมูลที่สะดวก ใช้งานง่าย จึงกลายเป็นตัวประสานมิดเดิลแวร์ ต่างๆ เข้าด้วยกันซึ่งจะให้ติดต่อกันเองคงทำได้ยาก เว็บได้ทำหน้าที่เป็นตัวกลางให้ มิดเดิลแวร์เพื่อสามารถคุยกันได้และมีประสิทธิภาพกว่าการใช้ มิดเดิลแวร์เพียงตัวใดตัวหนึ่ง

หากเรามองจากกรณีของเอ็นเทียร์ ( n-tier ) แอพพลิเคชัน จะพบว่า เว็บเซอร์วิสคือกลไกในการเข้าถึงบริการที่แต่ละมิดเดิลแวร์ให้บริการ การเข้าถึงจะอาศัย Listener และส่วนประกอบที่ระบุถึงบริการต่างๆ ที่รองรับการทำงาน โดยการทำงานจริงๆ นั้นก็ใช้วิธีการปกติของ มิดเดิลแวร์ นั้นๆ

พื้นฐานของเว็บเซอร์วิส คือ XML กับ HTTP ซึ่งจะพบว่า HTTP ก็เป็นที่รู้จักกันดี และไปได้ทั่วทุกแห่งที่มีอินเทอร์เน็ต ส่วน XML คือภาษาสากลที่สามารถอธิบายตนเองได้ เพื่อให้เกิดกิจกรรมระว่าง ไคลเอนต์ และบริการหรือระหว่างส่วนประกอบต่างๆ เบื้องหลังเว็บเซอร์วิสก็คือ ข้อความ XML จะถูกแปลงให้การขอบริการจากมิดเดิลแวร์ และผลที่ได้ก็จะแปลงกลับมาในรูป XML โดยมาตรฐาน XML นี้ทำให้เราไม่ต้องกังวลเรื่องของการเชื่อมโยงข้ามแพลตฟอร์มอีกต่อไป และโพรโตคอล ที่ส่งก็คือ HTTP นั่นเอง ถ้าท่านเชื่อมโยงกับ HTTP (หรือเว็บ) ได้ ท่านก็สามารถใช้งานเว็บเซอร์วิสได้นั่นเอง

แต่อย่างไรก็ตามการเข้าถึงเว็บเซอร์วิสนั้นยังเป็นเพียงโครงสร้างพื้นฐาน แต่ในความเป็นจริงยังมีอะไรมากกว่านั้น เช่น การค้นหาเซอร์วิส การทำธุระกรรม ความปลอดภัย การพิสูจน์ตน และอื่นๆ อันเป็นบริการที่ทำให้เป็นบริการพื้นฐานจริงๆ ซึ่งพื้นฐานของเว็บเซอร์วิส เต็มรูปแบบคือ XML + HTTP + SOAP + WSDL + UDDI หรือในระดับสูงกว่าเช่น WS-Security,Mobile Agent เป็นต้น สำหรับรายละเอียดในบางหัวข้อมีดังต่อไปนี้คือ

**1.2 วัตถุประสงค์ของโครงงาน**

ภายในโครงงานนี้ได้ทำการศึกษาถึงทฤษฎี การสร้าง และการนำ เว็บเซอร์วิสไปใช้งานโดยมีจุดประสงค์ดังนี้

1. ศึกษาเรื่องเว็บเซอร์วิส และเทคโนโลยีที่เกี่ยวข้อง ได้แก่ SOAP, XML, WSDL, UDDI เป็นต้น
2. ศึกษาการสร้างเว็บเซอร์วิส Bea Weblogic 8.14 ลงลึกในเทคโนโลยีของเว็บเซอร์วิส เช่น WS-Security , ความปลอดภัยในการเรียกใช้เว็บเซอร์วิส
3. ศึกษาวิธีการใช้งาน Bea Weblogic 8.14 เพื่อใช้สร้าง เว็บเซอร์วิส(Web Service)
4. ศึกษาการเรียกใช้เว็บเซอร์วิสข้ามค่ายรวมทั้งเปรียบเทียบเทคโนโลยีของทั้งสองค่าย.NET และจาวา

**1.3 ขอบเขตของโครงงาน**

1. ใช้ทฤษฎีของ WS-Security เพื่อเข้ามาเพิ่มความปลอดภัยให้กับเว็บเซอร์วิส (Web Service)
2. สร้างแอพพลิเคชั่นทำการเรียกใช้เว็บเซอร์วิสที่ใช้ WS-Security
3. พัฒนาเป็นระบบอีคอมเมิร์ซที่เกี่ยวกับ การจัดการจัดส่งสินค้า

**1.4 วิธีการดำเนินงานของโครงงาน**

1. ศึกษาทฤษฎีของ XML และ SOAP และส่วนประกอบในเว็บเซอร์วิส
2. ศึกษาทฤษฎีของ PKI และ SSL
3. ศึกษาทฤษฎีของ WS-Security
4. ศึกษาการพัฒนาเว็บเซอร์วิสใน Bea Weblogic 8.14
5. ศึกษาการปรับแต่งเพื่อให้เว็บเซอร์วิสมีความปลอดภัยในการสื่อสาร
6. ศึกษาการใช้งานร่วมกันรวมทั้งเปรียบเทียบเทคโนโลยีจากทั้งสองค่าย .NET และ จาวา

# บทที่ 2

# XML

## 2.1 แนะนำ XML

XML ย่อมาจาก eXtensible Markup Language XML ถูกออกแบบมาเพื่อใช้อธิบายข้อมูล ใน XML จะไม่มีแท็ก (tag) ที่กำหนดไว้ล่วงหน้า เราจะต้องกำหนดแท็กของเราขึ้นมาเอง และจะใช้ Document Type Definition (DTD) หรือ XML-Schema ในการอธิบายรูปแบบของข้อมูล เช่น

```
<BOOK>
   <TITLE>JAVA HOW TO PROGRAM</ TITLE >
   <ISDN>6-735645-23-7</ISDN>
   <PAGE>1447</PAGE>
<BOOK>
```

## 2.2 ไวยากรณ์ของ XML

ไวยากรณ์ของ XML ง่ายต่อการเรียนรู้และง่ายต่อการใช้งานจริง ด้วยเหตุนี้การสร้างซอฟต์แวร์ในการอ่านและจัดการกับข้อมูลจึงเป็นเรื่องง่าย ตัวอย่างของเอกสาร XML ดังนี้

```
<?XML version="1.0"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

ในบรรทัดแรกคือการประกาศเอกสาร XML ที่บอกถึง XML เวอร์ชัน โดยรูทอีลีเมนต์ (root element) คือ note โดย note มี 4 อีลีเมนต์ลูก (child element) คือ to,from,heading,body

กฎของเอกสาร XML มีดังนี้

- **XML อีลีเมนต์ (XML element)** ทุกตัวต้องมีการปิดแท็ก

```
<p>This is a paragraph</p>
```

- **XML แท็กมีความแตกต่างระหว่างตัวพิมพ์เล็กและตัวพิมพ์ใหญ่ (case sensitive)**

```
<message>This is correct</message>
```

- **XML อีลีเมนต์ทุกตัวต้องมีการซ้อนกันอย่างเหมาะสม**

```
<b><i>This text is bold and italic</i><b>
```

- **ทุกเอกสาร XML ต้องมีรูทแท็ก**

```
<root>
    <child>
        <subchild>.....</subchild>
    </child>
</root>
```

- **ค่าของแอตทริบิวต์ (Attribute) ต้องมีอยู่ภายในเครื่องหมายคำพูด (“”)**

```
<note date=“12/11/99”>
```

**2.3 XML อีลีเมนต์**

- **XML อีลีเมนต์สามารถขยายได้**

```
<note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
</note>
```

**และ สามารถเพิ่มข้อมูลเพิ่มเติมลงไปได้เช่น**

```
<note>
    <date>1999-08-01</date>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
</note>
```

- **XML อีลีเมนต์มีความสัมพันธ์กัน**

ในการที่จะเข้าใจถึงภาพรวมของ XML จะต้องเข้าใจถึงความสัมพันธ์ระหว่างชื่อของ XML อีลีเมนต์ที่ตั้ง และคอนเทนต์ (content) ของอีลีเมนต์

คำอธิบายของหนังสือดังนี้

```
Book Title: My First XML
Chapter 1: Introduction to XML
    What is HTML
    What is XML
Chapter 2: XML Syntax
    Elements must have a closing tag
    Elements must be correctly nested
```

สามารถมีเอกสาร XML ที่อธิบายหนังสือนั้น ได้ดังนี้

```
<book>
    <title>My First XML</title>
    <prod id="33-657" media="paper"></prod>
    <chapter>Introduction to XML
        <para>What is HTML</para>
        <para>What is XML</para>
    </chapter>
    <chapter>XML Syntax
        <para>Elements must have a closing tag</para>
```

```
    <para>Elements must be properly nested</para>
  </chapter>
</book>
```

- **อีลีเมนต์มีคอนเทนต์**

XML อีลีเมนต์คือทั้งหมดตั้งแต่เปิดแท็กจนถึงปิดแท็กอีลีเมนต์หนึ่งสามารถมีอีลีเมนต์คอนเทนต์ (element content) , มิกซ์คอนเทนต์ (mixed content) , ซิมเปิ้ลคอนเทนต์ (simple content) หรือ คอนเทนต์ว่างเปล่า (empty content) และยังมี แอตทริบิวต์ได้ด้วย

จากตัวอย่างข้างบน book มีอีลีเมนต์คอนเทนต์เพราะมันประกอบด้วยอีลีเมนต์อื่น chapter เป็น มิกซ์คอนเทนต์เพราะมันประกอบด้วยเท็กซ์และอีลีเมนต์อื่น ๆ para เป็นซิมเปิ้ลคอนเทนต์ หรือเท็กซ์คอนเทนต์ (text content) เพราะมันประกอบด้วยเท็กซ์เท่านั้น prod เป็นเอ็มตี้อีลีเมนต์เนื่องจากมันไม่มีข้อมูลใดๆ และเฉพาะ prod อีลีเมนต์เท่านั้นที่มีแอตทริบิวต์ แอตทริบิวต์ที่ชื่อ id มีค่าเท่ากับ “33-657” แอตทริบิวต์ที่ชื่อ media มีค่าเท่ากับ “paper”

**2.4 XML แอตทริบิวต์**

- **รูปแบบของโค้ท (Quote)**

```
<person sex=“female”>
<person sex=‘female’>
<gangster name='George "Shotgun" Ziegler'>
```

- **การใช้อีลีเมนต์และแอตทริบิวต์**

```
<person sex="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

**หรือ**

```
<person>
    <sex>female</sex>
    <firstname>Anna</firstname>
    <lastname>Smith</lastname>
</person>
```

- ปัญหาของการใช้แอตทริบิวต์
  - แอตทริบิวต์ไม่สามารถมีหลายค่า (อีลีเมนต์ย่อยสามารถมีได้)
  - แอตทริบิวต์ไม่ง่ายที่จะขยาย (ในอนาคต)
  - แอตทริบิวต์ไม่สามารถอธิบายโครงสร้าง (อีลีเมนต์ย่อยสามารถทำได้)
  - ค่าของแอตทริบิวต์ไม่ง่ายในการจะตรวจสอบกับ DTD ที่มีอยู่

แต่อย่างไรก็ตาม จะสามารถใช้แอตทริบิวต์ก็ต่อเมื่อ อินฟอร์เมชันนั้นไม่สัมพันธ์กับข้อมูล ตัวอย่างเช่น ในบางครั้ง ID อ้างอิง (reference) สามารถถูกใช้ในการเข้าถึง XML อีลีเมนต์

```
<messages>
    <note ID="501">
        <to>Tove</to>
        <from>Jani</from>
        <heading>Reminder</heading>
        <body>Don't forget me this weekend!</body>
    </note>
    <note ID="502">
        <to>Jani</to>
        <from>Tove</from>
        <heading>Re: Reminder</heading>
        <body>I will not!</body>
    </note>
</messages>
```

ID ในตัวอย่างนี้เป็นตัวนับ (counter) หรือค่าที่ไม่ซ้ำกัน (unique identifier) ในการอ้างถึง note ที่ต่างกันของไฟล์ XML ไม่ใช่ส่วนหนึ่งของข้อมูล note

**2.5 ความถูกต้อง XML (XML Validation)**

**1) "Well Formed" XML Document**

"Well Formed" XML document เป็นเอกสารที่เป็นไปตามกฎไวยากรณ์ XML ที่ได้กล่าวมาข้างต้น

**2) "Valid" XML Document**

"Valid" XML document เป็นเอกสาร XML ที่มีรูปแบบที่เป็นไปตามกฎของ DTD โดยจะต้องตรงตามโครงสร้างข้อมูลของเอกสารที่ได้ประกาศโดย DTD

- **XML DTD**

วัตถุประสงค์ของ DTD คือกำหนดโครงสร้างของเอกสาร XML เพื่อใช้ในการตรวจสอบว่าเอกสาร XML นั้นถูกต้องตามรูปแบบหรือไม่ เช่น element BOOK จะต้องมี element NAME เสมอ อาจจะมี element PAGE หรือไม่ก็ได้ เป็นต้น

- **โครงสร้างของ XML (XML Schema)**

โครงสร้างของ XML มีวัตถุประสงค์เหมือนกับ DTD จะถูกนำมาใช้แทน DTD เนื่องจาก

- ง่ายในการเรียนรู้มากกว่า DTD
- ใช้ได้ดีกว่า DTD
- โครงสร้าง (schema) เขียนด้วย XML
- รองรับดาต้าไทป์ (datatype)
- รองรับเนมสเปซ

**2.6 แนะนำ DTD**

**1) การประกาศ DOCTYPE ภายใน**

ถ้า DTD ถูกรวมอยู่ในไฟล์ XML นั้น มันจะต้องอยู่ภายในการให้คำจำกัดความ DOCTYPE ตามไวยากรณ์ นี้

```
<!DOCTYPE root-element [element-declarations]>
```

ตัวอย่างดังนี้

```
<?XML version="1.0"?>
<!DOCTYPE note [
  <!ELEMENT note (to,from,heading,body)>
  <!ELEMENT to     (#PCDATA)>
  <!ELEMENT from   (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
```

```
    <!ELEMENT body    (#PCDATA)>
]>
<note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend</body>
</note>
```

DTD ข้างบนนี้สามารถแปลความหมายได้ดังนี้

!DOCTYPE note (บรรทัดที่ 2) กำหนดว่าเป็นเอกสารของไทป์ (type) note

!ELEMENT note (บรรทัดที่ 3) กำหนดว่า note element มี 4 อีลิเมนต์ คือ to , from , heading และ body

!ELEMENT to (บรรทัดที่ 4) กำหนดว่า to อีลีเมนต์เป็นข้อมูลชนิด #PCDATA

!ELEMENT from (บรรทัดที่ 5) กำหนดว่า from อีลีเมนต์เป็นข้อมูลชนิด #PCDATA

**2) การประกาศ DOCTYPE ภายนอก**

ถ้า DTD อยู่ภายนอกไฟล์ XML มันจะต้องอยู่ภายในการให้คำจำกัดความ DOCTYPE ตาม ไวยากรณ์นี้

```
<!DOCTYPE root-element SYSTEM "filename">
```

ตัวอย่างไฟล์ XML เป็นดังนี้

```
<?XML version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

ไฟล์ของ DTD ชื่อ note.dtd เป็นดังนี้

**note.dtd**

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

**2.6.1 โครงสร้างของ XML**

เอกสาร XML จะถูกสร้างจากส่วนต่างๆ ดังนี้

- อีลีเมนต์ – เป็นโครงสร้างหลักของ XML
- แท็ก - ใช้ในการกำหนดมาร์กอัพอีลีเมนต์ (markup element)
- แอตทริบิวต์ - ให้ข้อมูลเพิ่มเติมกับอีลีเมนต์
- เอ็นติตี้ (entity) – เป็นตัวแปรที่ใช้ในการกำหนดเท็กซ์ทั่ว ๆ ไป เอ็นติตี้อ้างอิง (entity reference) เป็นการอ้างถึงเอ็นติตี้ เช่น เอ็นติตี้ชื่อ domain มีค่าเท่ากับ www.w3.org entity reference คือ &domain หากภายในเอกสาร XML มี &domain ค่าของมันจะมีค่าเท่ากับ www.w3.org
- PCDATA -หมายถึง parse character data PCDATA เป็นเท็กซ์ที่จะถูกแปล (parse) โดยพาร์เซอร์
- CDATA –หมายถึง character data เป็นเท็กซ์ที่จะไม่ถูกแปลโดยพาร์เซอร์

**2.6.2 การกำหนดอีลีเมนต์ใน DTD**

วิธีการในการประกาศอีลีเมนต์จะเป็นดังนี้

- **การประกาศอีลีเมนต์**

  ประกาศตามไวยากรณ์นี้

```
<!ELEMENT element-name category>
```

หรือ

```
<!ELEMENT element-name (element-content)>
```

- **อีลีเมนต์ว่างเปล่า (Empty)**

  ถูกประกาศโดยคีย์เวิร์ด EMPTY

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

**DTD example:**

<!ELEMENT br EMPTY>

**XML example:**

<br />

- **อีลีเมนต์ที่เป็นข้อมูลชนิดตัวอักษร**

  อีลีเมนต์ที่เป็นตัวอักษรเท่านั้นจะถูกประกาศด้วยคีย์เวิร์ด PCDATA ที่เปิดและปิดด้วย "(" ")"

<!ELEMENT element-name (#PCDATA)>

**DTD example:**

<!ELEMENT note (#PCDATA)>

- **อีลีเมนต์ที่เป็นข้อมูลใด ๆ**

  ประกาศกับคีย์เวิร์ด ANY คอนเท็นต์จะสามารถเป็นข้อมูลชนิดใดก็ได้

<!ELEMENT element-name ANY>

**DTD example:**

<!ELEMENT note ANY>

- **อีลีเมนต์และอีลีเมนต์ลูก (Sequences)**

  ประกาศโดยใช้ "," ในการแยกอีลิเมนต์ลูกแล้วเปิดและปิดด้วย "(" ")"

<!ELEMENT element-name (child-element-name)>

หรือ

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

**DTD example:**

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

**ถ้ามีการประกาศอีลีเมนต์ลูกจะต้องประกาศอีลีเมนต์ลูกให้สมบูรณ์ด้วยดังตัวอย่างข้างล่างนี้**

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

- **การประกาศเพียงครั้งเดียวของอีลีเมนต์เดียวกัน**

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

**DTD example:**

```
<!ELEMENT note (message)>
```

จากตัวอย่างหมายความว่าอีลีเมนต์ note จะประกอบด้วยอีลีเมนต์ลูก message 1 ค่าเท่านั้น

- **การประกาศอย่างน้อย 1 ครั้งของอีลีเมนต์เดียวกัน**

```
<!ELEMENT element-name (child-name+)>
```

**DTD example:**

```
<!ELEMENT note (message+)>
```

เครื่องหมาย + ถูกใช้ในความหมายหนึ่งหรือมากกว่า จึงหมายความว่าอีลีเมนต์ note จะประกอบด้วยอีลีเมนต์ลูก message ตั้งแต่ 1 ค่าขึ้นไป

- **การไม่ประกาศหรือประกาศเท่าไรก็ได้ของอีลีเมนต์เดียวกัน**

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

**DTD example:**

```
<!ELEMENT note (message*)>
```

เครื่องหมาย * ถูกใช้ในความหมายว่าไม่มีหรือมีเท่าไรก็ได้ จึงหมายความว่าอีลีเมนต์ note จะประกอบด้วยอีลีเมนต์ลูก message กี่ค่าก็ได้ หรือไม่มีเลยก็ได้

- การไม่ประกาศหรือประกาศเพียงครั้งเดียวของอีลีเมนต์เดียวกัน

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

**DTD example:**

```
<!ELEMENT note (message?)>
```

เครื่องหมาย ? ถูกใช้ในความหมายไม่มีหรือมีเพียงหนึ่ง จึงหมายความว่าอีลีเมนต์ note จะประกอบด้วยอีลีเมนต์ลูก message หนึ่งค่า หรือไม่มีก็ได้

- การประกาศคอนเท็นต์แบบให้เลือก

  จะใช้เครื่องหมาย | แทนความหมายหรือ

**DTD example:**

```
<!ELEMENT note (to,from,header,message|body)>
```

ดังนั้นตัวอย่างนี้จึงหมายถึงอีลีเมนต์ note จะต้องประกอบด้วยอีลีเมนต์ to, อีลีเมนต์ from, อีลีเมนต์ header และอีลีเมนต์ message หรืออีลีเมนต์ body อย่างใดอย่างหนึ่งเท่านั้น

- การประกาศคอนเท็นต์แบบผสม

  เป็นการผสมการประกาศหลาย ๆ อย่างเข้าด้วยกัน

**DTD example:**

```
<!ELEMENT note (#PCDATA|to|from|header|message)*>
```

ตัวอย่างนี้หมายถึงอีลีเมนต์ note จะประกอบด้วยตัวอักษรที่ไม่ถูกแปลโดยพาร์เซอร์ หรืออีลีเมนต์ to , from , header , และ message จำนวนเท่าไรก็ได้

**2.6.3 การกำหนดแอตทริบิวต์ใน DTD**

วิธีการประกาศแอตทริบิวต์จะเป็นดังนี้

- การประกาศแอตทริบิวต์

  แอตทริบิวต์ต้องถูกประกาศตามไวยากรณ์ นี้

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

**DTD example:**

<!ATTLIST payment type CDATA "check">

**XML example:**

<payment type="check">

ชนิดของแอตทริบิวต์สามารถมีค่าได้ดังตารางที่ 2-1

| ค่า | คำอธิบาย |
|---|---|
| CDATA | ค่าต้องเป็นตัวอักษรที่จะไม่ถูกแปลโดยพาร์เซอร์ |
| (en1\|en2\|..) | ค่าต้องเป็นอันใดอันหนึ่งในรายการนั้น |
| ID | ค่าต้องเป็น id ที่ไม่ซ้ำกัน |
| IDREF | ค่าต้องเป็น id ของอีลีเมนต์อื่น |
| IDREFS | ค่าต้องเป็นรายการของ id อื่น โดยจะเว้นแต่ละรายการด้วยช่องว่าง |
| NMTOKEN | ค่าต้องเป็นชื่อที่ประกอบด้วยตัวอักษร , ตัวเลข , เครื่องหมายจุด (.) , เครื่องหมายขีดกลาง (-) , เครื่องหมายขีดล่าง (_) และสามารถมีเครื่องหมายโคลอน (:) ยกเว้นในตำแหน่งแรกได้ |
| NMTOKENS | ค่าต้องเป็นรายการของ NMTOKEN โดยจะเว้นแต่ละรายการด้วยช่องว่าง |
| ENTITY | ค่าต้องเป็นเอ็นติตี้ |
| ENTITIES | ค่าต้องเป็นรายการของเอ็นติตี้ |
| NOTATION | ค่าต้องเป็นชื่อของ notation (notation คือชนิดของข้อมูลภายนอกที่กำหนดขึ้นเอง เช่น GIF เพื่อบอกว่าเป็นไฟล์รูปภาพ) |
| XML: | ค่าต้องขึ้นต้นด้วย XML |

*ตารางที่ 2-1 ชนิดของ แอตทริบิวต์ใน XML*

ค่าดีฟอลต์ (default-value) สามารถมีค่าได้ดังตารางที่ 2-2

| ค่า | คำอธิบาย |
|---|---|
| #DEFAULT value | ค่าดีฟอลต์ของแอตทริบิวต์ |
| #REQUIRED | อีลีเมนต์ต้องมีแอตทริบิวต์ |
| #IMPLIED | อีลีเมนต์จะมีแอตทริบิวต์หรือไม่ก็ได้ |
| #FIXED value | แอตทริบิวต์ต้องเหมือนกับที่กำหนดไว้ |

*ตารางที่ 2-2 ค่าดีฟอลต์ของ แอตทริบิวต์ใน XML*

- ตัวอย่างของการประกาศ แอตทริบิวต์

**DTD example:**

```
<!ELEMENT square EMPTY>
<!ATTLIST square width CDATA "0">
```

**XML example:**

```
<square width="100"></square>
```

ตัวอย่างนี้หมายความว่าสแควร์อีลีเมนต์ (square element) เป็นอีลีเมนต์ว่างกับแอตทริบิวต์ width ซึ่งเป็นชนิดซีดาต้า ถ้าไม่ระบุจะยึดค่าดีฟอลต์เป็น 0

- ค่าของ แอตทริบิวต์แบบดีฟอลต์

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

**DTD example:**

```
<!ATTLIST payment type CDATA "check">
```

**XML example:**

```
<payment type="check">
```

ถ้าในเอกสาร XML ไม่มีการระบุจะใช้ค่าดีฟอลต์ แต่ถ้าระบุก็จะใช้ตามที่ระบุไว้

- แอตทริบิวต์โดยนัย

```
<!ATTLIST element-name attribute-name attribute-type #IMPLIED>
```

**DTD example:**

<!ATTLIST contact fax CDATA #IMPLIED>

**XML example:**

<contact fax="555-667788">

จะใช้ #IMPLIED ในกรณีจะระบุแอตทริบิวต์หรือไม่ก็ได้ ถ้าไม่ระบุก็จะไม่มีค่าดีฟอลต์ให้

- แอตทริบิวต์ที่ต้องระบุเสมอ

<!ATTLIST element-name attribute_name attribute-type #REQUIRED>

**DTD example:**

<!ATTLIST person number CDATA #REQUIRED>

**XML example:**

<person number="5677">

ใช้ในการบังคับให้ต้องระบุแอตทริบิวต์ถ้าในเอกสาร XML ไม่ระบุ แอตทริบิวต์เอกสารนั้นจะไม่ถูกต้อง

- ค่าของแอตทริบิวต์คงที่

<!ATTLIST element-name attribute-name attribute-type #FIXED "value">

**DTD example:**

<!ATTLIST sender company CDATA #FIXED "Microsoft">

**XML example:**

<sender company="Microsoft">

ในแบบนี้ถ้าไม่ได้ระบุค่าแอตทริบิวต์จะใช้ค่าดีฟอลต์ แต่ถ้าจะระบุต้องระบุให้ตรงกับค่าดีฟอลต์ มิฉะนั้นจะถือว่าผิด

- ค่าของแอตทริบิวต์กำหนดเอง

<!ATTLIST element-name attribute-name (en1|en2|..) default-value>

**DTD example:**

```
<!ATTLIST payment type (check|cash) "cash">
XML example:
<payment type="check">
หรือ
<payment type="cash">
```

จะใช้เมื่อเราต้องการกำหนดค่าของแอตทริบิวต์เอง

**2.6.4 การกำหนดเอ็นติตี้ใน DTD**

- การประกาศเอ็นติตี้ภายใน

```
<!ENTITY entity-name "entity-value">
```

```
DTD Example:
<!ENTITY writer "Donald Duck.">
<!ENTITY copyright "Copyright W3Schools.">
XML example:
<author>&writer;&copyright;</author>
```

- การประกาศเอ็นติตี้ภายนอก

```
<!ENTITY entity-name SYSTEM "URI/URL">
```

```
DTD Example:
<!ENTITY writer
 SYSTEM "http://www.w3schools.com/entities/entities.XML">
<!ENTITY copyright
 SYSTEM "http://www.w3schools.com/entities/entities.dtd">
XML example:
<author>&writer;&copyright;</author>
```

**2.7 XML เนมสเปซ**

XML เนมสเปซจะหาวิธีหลีกเลี่ยงความสับสนที่เกิดจากชื่ออิลิเมนต์ (Name Conflicts)

- **ความสับสนที่เกิดจากชื่อ**

ความสับสนที่เกิดจากชื่ออีลีเมนต์จะเกิดขึ้นบ่อยเมื่อเอกสารที่แตกต่างกันใช้ชื่อเดียวกันในการอธิบายชนิดของอีลีเมนต์ที่แตกต่างกัน เช่น ตารางใน 2 ตัวอย่างต่อไปนี้

```
<table>
  <tr>
  <td>Apples</td>
  <td>Bananas</td>
  </tr>
</table>
```

และ

```
<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

ถ้านำเอกสาร XML ทั้ง 2 มารวมกัน จะเกิดความสับสนจากชื่อขึ้นเพราะทั้ง 2 มีอีลีเมนต์ (table)ซึ่งมีใช้ในความหมายที่ต่างกัน

- **การแก้ปัญหาความสับสนที่เกิดจากชื่อโดยใช้คำเติมหน้า (Prefix)**

เอกสาร XML ทั้ง 2 คือ

```
<h:table>
  <h:tr>
  <h:td>Apples</h:td>
  <h:td>Bananas</h:td>
  </h:tr>
</h:table>
```

และ

```
<f:table>
  <f:name>African Coffee Table</f:name>
```

```
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

ความสับสนที่เกิดจากชื่อจะถูกกำจัดไปโดยการเพิ่มคำเติมหน้าทำให้ได้อีลีเมนต์ที่ต่างกัน 2 ชนิด (<h:table> และ <f:table>)

- **การใช้เนมสเปซ**

  จากเอกสาร XML ทั้ง2 คือ

```
<h:table xmlns:h="http://www.w3.org/TR/html4/">
  <h:tr>
  <h:td>Apples</h:td>
  <h:td>Bananas</h:td>
  </h:tr>
</h:table>
```

และ

```
<f:table xmlns:f="http://www.w3schools.com/furniture">
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

เราสามารถเพิ่มแอตทริบิวต์ xmlns ลงในแท็ก เพื่อบอกชื่อที่ระบุไว้ ที่เกี่ยวข้องกับเนมสเปซ

- **แอตทริบิวต์เนมสเปซ**

  แอตทริบิวต์เนมสเปซจะวางในแท็กเริ่มต้น (start tag) ตามไวยากรณ์ ดังนี้

```
Xmlns:namespace-prefix="namespace"
```

จากตัวอย่างข้างบนเนมสเปซจะสามารถกำหนดด้วยที่อยู่ในอินเตอร์เน็ต (Internet address)

```
Xmlns:f="http://www.w3schools.com/furniture">
```

จากเนมสเปซเจาะจง (namespace specification) ของ W3C กล่าวไว้ว่าเนมสเปซสามารถเป็น Uniform Resource Identifier (URI)

ถ้าเนมสเปซถูกกำหนดในแท็กเริ่มต้นแล้วอีลีเมนต์ลูกทุกตัวที่มีคำเติมหน้าเหมือนกันจะต้อง มีความสัมพันธ์กับเนมสเปซเดียวกันด้วย

และที่อยู่ที่ใช้บอกเนมสเปซใช้เพื่อให้ชื่อมีความเป็นเอกลักษณ์ไม่ได้ใช้เป็นที่อยู่เพื่ออ้างอิงหาข้อมูล แต่หลายบริษัทใช้เนมสเปซเพื่อเป็นตัวชี้ไปยังหน้าเว็บที่มีอยู่จริงซึ่งเก็บข้อมูลเกี่ยวกับเนมสเปซไว้

- **ดีฟอลต์เนมสเปซ (Default Namespaces)**

การกำหนดดีฟอลต์เนมสเปซจะป้องกันการใช้คำเติมหน้าในอีลีเมนต์ลูกทุกตัว มีไวยากรณ์ดังนี้

```
<element xmlns="namespace">
```

ซึ่งจากตัวอย่างที่ผ่านมาจะได้ดังนี้

```
<table xmlns="http://www.w3.org/TR/html4/">
  <tr>
  <td>Apples<td>
  <td>Bananas<td>
  <tr>
</table>
```

และ

```
<table xmlns="http://www.w3schools.com/furniture">
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

## 2.8 XML PCDATA และ CDATA

Parsed Character Data (PCDATA) คือข้อมูลแบบเท็กซ์ที่จะถูกแปลโดยพาร์เซอร์

Character Data (CDATA) คือข้อมูลแบบเท็กซ์ที่ไม่ถูก parse โดยพาร์เซอร์

### 2.8.1 PCDATA

- **XML พาร์เซอร์จะมองทุกเท็กซ์เป็น Parsed Characters (PCDATA)**

เมื่อ XML ถูกแปลเท็กซ์ที่อยู่ระหว่าง XML แท็กจะถูกแปลไปด้วย

```
<message>This text is also parsed</message>
```

พาร์เซอร์ทำเช่นนี้เนื่องจาก XML อีลีเมนต์สามารถมีหลายอีลีเมนต์อยู่ภายใน ดังในตัวอย่าง

```
<name><first>Bill</first><last>Gates</last></name>
```

และพาร์เซอร์จะแบ่งเป็นซับอีลีเมนต์ (sub-element) ดังนี้

```
<name>
  <first>Bill</first>
  <last>Gates</last>
</name>
```

- **ตัวอักษรเอสเคป (Escape Character)**

ตัวอักษร XML ที่ไม่สามารถใช้ได้จะถูกแทนที่ด้วยเอ็นติตี้อ้างอิง

ถ้าใส่ตัวอักษร "<" ใน XML อีลีเมนต์จะทำให้เกิดความผิดพลาด เพราะพาร์เซอร์จะแปลเป็นจุดเริ่มต้นของอีลีเมนต์ใหม่ ดังนั้นจึงไม่สามารถเขียน อย่างนี้ได้

```
<message>if salary < 1000 then</message>
```

การหลีกเลี่ยง สามารถทำได้โดย แทนที่ "<" ด้วยเอ็นติตี้อ้างอิงดังนี้

```
<message>if salary &lt; 1000 then</message>
```

เอ็นติตี้อ้างอิงมาตรฐานที่ได้กำหนดมาให้ก่อนแล้วใน XML มีทั้งหมด 5 ตัว ดังตารางที่ 2-3

| Entity Reference | ค่า | ความหมาย |
|---|---|---|
| < | < | น้อยกว่า |
| > | > | มากกว่า |
| & | & | ampersand |
| ' | ' | apostrophe |
| " | " | quotation mark |

***ตารางที่ 2-3 เอ็นติตี้อ้างอิงมาตรฐาน***

เอ็นติตี้อ้างอิงจะขึ้นต้นด้วย & และลงท้ายด้วย ;

เฉพาะ "<" และ "&" เท่านั้นที่ไม่สามารถใช้ได้ใน XML ส่วนตัวที่เหลือควรทำให้เป็นนิสัย

**2.8.2 CDATA**

พาร์เซอร์จะไม่แปลทุกอย่างที่อยู่ในส่วนของ CDATA

ถ้าเท็กซ์มี "<" หรือ "&" อยู่ภายใน XML element นั้นจะถูกกำหนดเป็นส่วนของ CDATA ส่วนของ CDATA จะเริ่มต้นด้วย "<![CDATA[" และลงท้ายด้วย "]]>"

```
<script>
<![CDATA[
function matchwo(a,b)
{
if (a < b && a < 0) then
  {
  return 1
  }
else
  {
  return 0
  }
}
]]>
</script>
```

จากตัวอย่างข้างต้น ทุกอย่างที่อยู่ในส่วนของ CDATA จะถูกเพิกเฉยโดยพาร์เซอร์

**2.9 XML พาร์เซอร์**

XML พาร์เซอร์คือโปรแกรมที่ใช้ในการอ่าน สร้าง และจัดการกับเอกสาร XML รวมถึงตรวจความถูกต้องของเอกสาร XML พาร์เซอร์สามารถเขียนได้ด้วยภาษาใด ๆ ก็ได้ ในบราวเซอร์ IE 5.0 ขึ้นไปก็มี Microsoft XML parser ทำให้ IE สามารถแสดงผลเอกสาร XML ได้ XML พาร์เซอร์ของภาษาจาวาอย่างเช่น XML พาร์เซอร์ของซัน (http://java.sun.com/xml) หากเราต้องการสร้างแอพพลิเคชันที่สามารถจัดการกับ XML เราก็ต้องมีพาร์เซอร์ของภาษานั้นและเขียนโค้ดติดต่อกับ API ของมัน

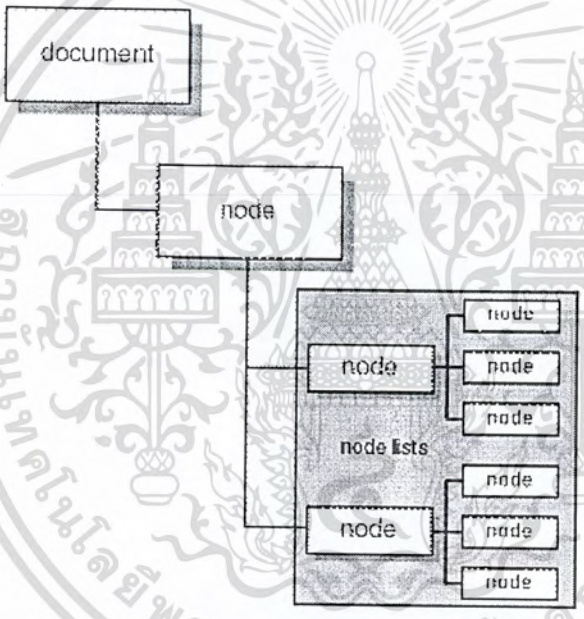
XML พาร์เซอร์มี 2 ชนิดคือ

- DOM (Document Object Model)
- SAX (Simple API for XML)

### 2.9.1 DOM

DOM จะใช้วิธีแปลเอกสาร XML เป็นลำดับชั้น (hierachy) ของออบเจ็กต์ดังนี้ (ดังรูปที่ 2-1)

1) เอกสารอ็อบเจ็ค (Document Object) – แหล่งข้อมูล XML
2) โหนดอ็อบเจ็ค (Node Object) – โหนดพ่อ(Parent node) ของโหนดลูก (child node)
3) โหนดลิสต์อ็อบเจ็ค (Node List Object) – ลิสต์ของ sibling node
4) แปลผิดพลาด (Parse Error) – ใช้ในการรับค่าผิดพลาดที่เกิดจากการแปล

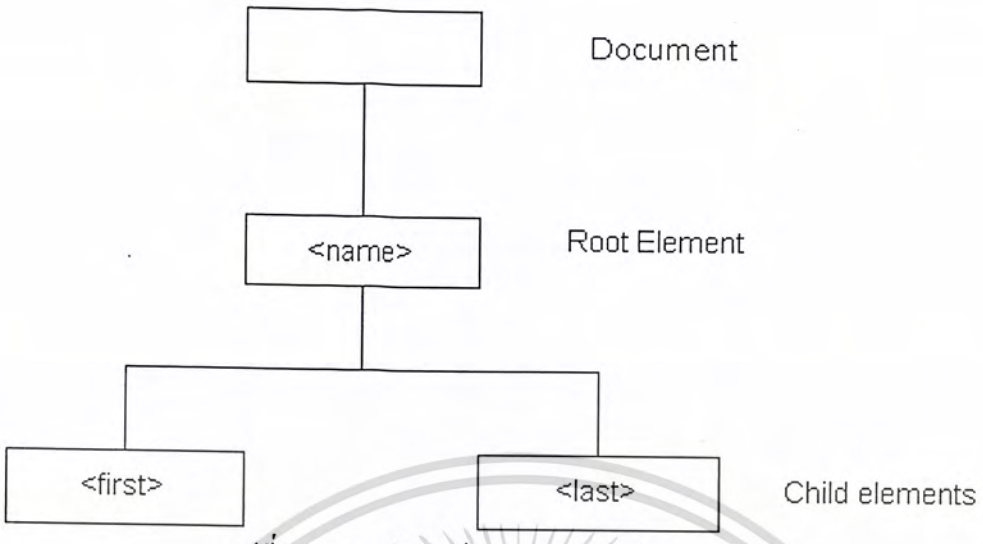



*รูปที่ 2-1 โครงสร้างของ DOM*

หากเอกสาร XML เป็นดังนี้

```
<name>
   <first>Jim(/first>
   <last>Jones</last>
</name>
```

จะได้ความสัมพันธ์ดังรูปที่ 2-2




*รูปที่ 2-2 ความสัมพันธ์ของการแปลด้วย DOM*

**2.9.2 SAX**

SAX (Simple API for XML) วิธีนี้จะเป็นแบบตอบสนองเหตุการณ์ event-driven โดยเหตุการณ์จะประกอบด้วย

1) อีลีเมนต์เริ่มต้น (Start Element) – จะตอบสนองกับเหตุการณ์นี้เมื่อพบจุดเริ่มต้นของแท็ก XML
2) อีลีเมนต์สิ้นสุด (End element) – จะตอบสนองกับเหตุการณ์นี้เมื่อพบจุดสิ้นสุดของแท็ก
3) ตัวอักษร (Character) – จะตอบสนองกับเหตุการณ์นี้เมื่อพบกับตัวอักษร
4) เริ่มต้นเอกสาร (Start Document) – จะตอบสนองกับเหตุการณ์นี้ในตอนเริ่มแปล
5) สิ้นสุดเอกสาร (End Document) - จะตอบสนองกับเหตุการณ์นี้ในเมื่อจบการแปล

ดังนั้นจากโค้ดข้างบนจะเกิดเหตุการณ์ดังนี้

| | | |
|---|---|---|
| | → | เริ่มต้นเอกสาร |
| <name> | → | อีลีเมนต์เริ่มต้น name |
| <first> | → | อีลีเมนต์เริ่มต้น first |
| Jim | → | ตัวอักษร Jim |
| </first> | → | อีลีเมนต์สิ้นสุด first |
| <last> | → | อีลีเมนต์เริ่มต้น last |
| Jones | → | ตัวอักษร Jones |
| </last> | → | อีลีเมนต์สิ้นสุด last |
| </name> | → | อีลีเมนต์สิ้นสุด name |
| | → | สิ้นสุดเอกสาร |

ทั้ง 2 วิธีมีข้อดีข้อเสียต่างกันคือ DOM จะใช้เมโมรีและมีการประมวลผลมาก แต่สามารถจัดการได้ง่าย สามารถที่จะแก้ไขค่าได้ ไม่เหมาะกับเอกสาร XML ที่มีขนาดใหญ่ ถูกนำไปใช้ในบราวเซอร์ ในขณะที่ SAX ไม่ต้องแปลเอกสาร XML ทั้งหมดจึงใช้โมเมรีและมีการประมวลผลน้อยกว่า เหมาะกับการสร้างและรับค่าจากเอกสาร จะถูกใช้ในการประมวลผลทางฝั่งเซิร์ฟเวอร์ ตัวอย่าง XML พาร์เซอร์ดังตารางที่ 2-4

| ชื่อ | แพล็ตฟอร์ม | คำบรรยาย |
|---|---|---|
| ActiveDOM | Microsoft Window | http://www.vivid-creations.com |
| ActiveSAX | Microsoft Window | http://www.vivid-creations.com |
| JavaTM API for XML | Java | http://java.sun.com/xml |
| MS XML | Microsoft Window | http://www.microsoft.com |
| Oracle XML Developer's Kit | Java , C++ , PL/SQL | http://technet.oracle.com/tech/xml |
| Xerces | Java , C++ | http://xml.apache.org |
| XP | Java | http://www.jclark.com/xml |

*ตารางที่ 2-4 XML พาร์เซอร์*

**2.10 ประโยชน์ของ XML**

สำหรับประโยชน์ของ XML นั้น เป็นด้านความยืดหยุ่นในการใช้งานสำหรับแอพพลิเคชันที่อิงกับเว็บที่ใช้ง่ายในการค้นหาข้อมูล มีความยืดหยุ่นในการพัฒนาเว็บ สามารถผสมผสานข้อมูลจากหลายแหล่งจากแอพพลิเคชันที่ต่างกัน สามารถแสดงข้อมูลแบบต่างๆ และสามารถปรับปรุงข้อมูลให้ทันสมัยเสมอ และคาดว่าจะเป็นมาตรฐานใหม่ของระบบเปิด ซึ่งนับเป็นรูปแบบใหม่สำหรับการส่งข้อมูลบนเว็บที่มากด้วยข้อมูลหลายแบบ แต่ส่งผ่านด้วยเทคโนโลยีที่บีบอัดข้อมูลที่ให้ความเร็วได้รับการสนับสนุนจากผลิตภัณฑ์ค่ายไมโครซอฟท์

สถาปัตยกรรม XML

ภาษา XML ได้รับการสนับสนุนจาก W3C ให้นักพัฒนาเว็บแอพพลิเคชันได้หันมาใช้เป็นส่วนประกอบของการพัฒนาเว็บไซต์ เพราะ XML มีประสิทธิภาพและมีความน่าเชื่อสูงในการแปลงข้อมูลและโครงสร้างข้อมูลให้สามารถนำไปใช้งาน สถาปัตยกรรม 3 เทียร์ ที่ XML สามารถสร้างขึ้นจากระบบข้อมูลที่ใช้โมเดลของ 3-เทียร์ โครงสร้าง ของข้อมูลต่างๆ สามารถนำมาแสดงตามข้อกำหนด หรือรูปแบบที่ต้องการตามการใช้งานได้

**2.11 บทสรุปของ XML**

XML มีความยืดหยุ่นพอในการนำเสนอข้อมูลแบบต่างๆ ที่สามารถให้รายละเอียดโครงสร้างข้อมูลตามระดับและความต้องการในการนำไปใช้งาน

# บทที่ 3

# SOAP + WSDL

## 3.1 SOAP

### 3.1.1 แนะนำ SOAP

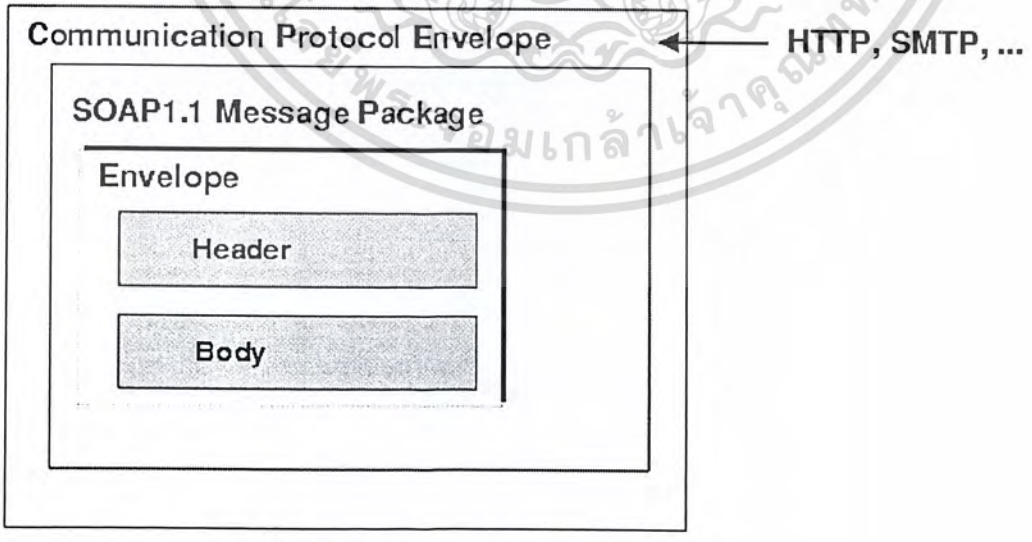
SOAP (Simple Object Access Protocol) เป็นโพรโตคอลที่ใช้ XML เป็นพื้นฐาน เพื่อให้ซอฟต์แวร์คอมโพเนนต์ และแอพพลิเคชันสามารถติดต่อกันผ่าน HTTP ซึ่งเป็นมาตรฐานอินเตอร์เน็ตโพรโตคอลได้

เนื่องจากการสื่อสารระหว่างแอพพลิเคชันจำเป็นต่อการพัฒนาแอพพลิเคชันบนอินเตอร์เน็ต เป็นอย่างมาก แต่แอพพลิเคชันแบบกระจายกันทำงาน (distributed application) ในปัจจุบันใช้ Remote Procedure Call (RPC) สื่อสารกันระหว่างออบเจ็กต์ เช่น DCOM และ CORBA ซึ่งไม่ได้ใช้พอร์ตเดียวกับ HTTP ที่เป็นพอร์ตมาตรฐานสำหรับให้บริการเว็บ ดังนั้น RPC จึงนำมาปรับใช้กับอินเตอร์เน็ตได้ยาก และมีปัญหาทางด้านความปลอดภัย ไฟร์วอลล์และพร็อกซีเซิร์ฟเวอร์จะไม่ยอมให้ส่งข้อมูลชนิดนี้ได้ตามปกติ วิธีที่ดีกว่าคือใช้ HTTP เพราะเป็นที่ยอมรับโดยอินเตอร์เน็ตบราวเซอร์ และเซิร์ฟเวอร์ทุกชนิด ซึ่ง SOAP ถูกสร้างมาเพื่อใช้ในกรณีนี้

ข้อดีของ SOAP ก็คือ SOAP ไม่ขึ้นกับคอมโพเนนต์เทคโนโลยี และภาษาการเขียนโปรแกรมใดๆ สามารถเขียนได้ง่ายและขยายเพิ่มเติมได้

### 3.1.2 ส่วนประกอบของ SOAP

SOAP เมสเซจใช้ไวยากรณ์ของ XML ในการสร้าง ประกอบด้วย 3 อีลีเมนต์มาตรฐาน คือ SOAP Envelope, SOAP Header และ SOAP Body




*รูปที่ 3-1 โครงสร้างของ SOAP*

SOAP เมสเซจ จะต้องเป็นไปตามกฎนี้

- Envelope เป็นอีลีเมนต์ที่อยู่บนสุด ต้องมีอีลีเมนต์นี้เสมอ
- Header อาจจะมีหรือไม่มีก็ได้ แต่ถ้ามีต้องเป็นอีลีเมนต์ลูกอีลีเมนต์แรกของenvelope
- Body ต้องเป็นอีลีเมนต์ลูกอีลีเมนต์แรกของenvelopeหรือheader

SOAP ร้องขอจะเป็นดังนี้

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
<env:Body>
<n1:plus xmlns:n1="http://www.openuri.org/">
<n1:i xmlns:n1="http://www.openuri.org/">5</n1:i>
<n1:j xmlns:n1="http://www.openuri.org/">6</n1:j>
</n1:plus>
</env:Body>
</env:Envelope>
```

SOAP ตอบสนองจะเป็นดังนี้

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
<SOAP-ENV:Body>
<ns:plusResponse xmlns:ns="http://www.openuri.org/">
<ns:plusResult>11</ns:plusResult>
</ns:plusResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

นอกจากนั้นใน SOAP เมสเซจ จะมีการใช้ XML เนมสเปซ ทุกๆ อีลีเมนต์ในเอกสารจะขึ้นต้นด้วยเนมสเปซ เนมสเปซจะถูกกำหนดโดยใช้ xmlns แอตทริบิวต์ดังนี้

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/" >
```

แต่หากไม่ต้องการให้ขึ้นต้นด้วยเนมสเปซก็สามารถทำได้ ดังนี้

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/" >
```

**3.1.2.1 Envelope**

envelope อีลีเมนต์เป็นอีลีเมนต์บนสุดของเอกสาร XML ที่ใช้แสดงเมสเซจ อาจประกอบด้วยแอตทริบิวต์ คือ envelope เนมสเปซ และ encodingStyle

1. envelope เนมสเปซ – SOAP เมสเซจจะแสดงเวอร์ชันโดยใช้ เนมสเปซ ของ envelope อีลีเมนต์ เนมสเปซ จะถูกนำไปใช้ขึ้นต้น envelope, headerและ Body อีลีเมนต์
2. EncodingStyle แอตทริบิวต์ – ใช้แสดงวิธี serialization ของ SOAP เมสเซจ แอตทริบิวต์นี้สามารถมีได้ในทุกอีลีเมนต์และจะมีผลกับ คอนเท็นต์ของอีลีเมนต์นั้นและอีลีเมนต์ลูกทั้งหมดของมันที่ไม่ได้ประกาศแอตทริบิวต์

**3.1.2.2 Body**

เป็นส่วนของข้อมูลที่ใช้ในการแลกเปลี่ยน โดยทั่ว ๆ ไปข้อมูลจะเป็นการ marshall RPC call และ error report อีลีเมนต์ลูกของBody อีลีเมนต์จะถูกเรียกว่า Body entry

Body entry จะต้องเป็นไปตามกฎนี้

- Body entry จะต้องแสดงด้วยชื่อเต็มที่ประกอบด้วย เนมสเปซ URI และ ชื่อของมัน (local name)
- SOAP encodingStyle แอตทริบิวต์อาจจะมีได้ เพื่อแสดงถึงวิธีเข้ารหัส (encode) ของ Body entry นั้น

**3.1.2.3 header (Header)**

เป็นส่วนเพิ่มเติมของเมสเซจสามารถประกอบด้วยอินฟอร์เมชันที่ระบุไปยังแอพพลิเคชัน ส่วนเพิ่มเติมนี้อาจนำไปใช้อิมพลีเม็นต์เป็น authentication, transaction management เป็นต้น ทุก ๆ อีลีเมนต์ลูกของ header อีลีเมนต์จะถูกเรียกว่า Header entry

Header entry จะต้องเป็นไปตามกฎดังนี้

- Header entry จะต้องแสดงด้วยชื่อเต็มที่ประกอบด้วย เนมสเปซ URI และ ชื่อของมัน (local name)
- อาจมี SOAP encodingStyle แอตทริบิวต์ที่ใช้สำหรับHeader entry
- SOAP mustUnderStand แอตทริบิวต์และ SOAP actor แอตทริบิวต์จะมีหรือไม่ก็ได้ เพื่อใช้บอกว่าจะจัดการกับเอ็นทรีนั้นอย่างไรและโดยใคร

### 3.1.3 SOAP ฟอลต์อีลีเมนต์ (SOAP Fault Element)

ข้อความแสดงความผิดพลาดจากแอพพลิเคชันของ SOAP จะเก็บอยู่ในฟอลต์อีลีเมนต์ ซึ่งถ้ามีจะต้องปรากฏใน body อีลีเมนต์เพียงครั้งเดียวใน SOAP เมสเซจ ตัวอย่างอาจเป็นดังนี้

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
        soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <soap:Body>
    <soap:Fault>
      <faultcode>soap:MustUnderstand</faultcode>
      <faultstring>Mandatory Header error.</faultstring>
      <faultactor>http://www.wrox.com/heroes/endpoint.asp</faultactor>
      <detail>
        <w:source xmlns:w="http://www.wrox.com/">
          <module>endpoint.asp</module>
          <line>203</line>
        </w:source>
      </detail>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

SOAP ฟอลต์อีลีเมนต์มีอีลีเมนต์ย่อย ๆ ดังตารางที่ 2-5

| Sub Element | Description |
|---|---|
| <faultcode> | โค้ดที่ระบุถึงการ error |
| <faultstring> | ข้อความการ error |
| <faultactor> | ใครเป็นสาเหตุของการ error |
| <detail> | ระบุอินฟอร์เมชันของการ error |

*ตารางที่ 3-1 SOAP Fault Element*

ค่าของฟอลต์โค้ด (faultcode) สามารถมีค่าได้ดังตารางที่ 2-6

| Error | Description |
|---|---|
| VersionMismatch | เนมสเปซ ภายใน SOAP เอ็นเวโลปอีลีเมนต์ไม่ถูกต้อง |
| MustUnderstand | อีลีเมนต์ลูกของเฮดเดอร์อีลีเมนต์กับ mustUnderstand แอตทริบิวต์ที่มีค่า "1" ผู้รับไม่รองรับ |
| Client | เมสเซจมีรูปแบบไม่ถูกต้อง หรือมีอินฟอร์เมชันที่ไม่ถูกต้อง |
| Server | เกิดปัญหากับเซิร์ฟเวอร์ ไม่สามารถประมวลผลได้ |

*ตารางที่ 3-2 ค่าฟอลต์โค้ด*

### 3.1.4 SOAP เอ็นโค้ดดิ้ง

SOAP เอ็นโค้ดดิ้งมีวิธีการ map จากไทป์ของโปรแกรมมิ่งไปเป็น XML 2 วิธี คือจากภายนอกโดยใช้ WSDL (Web Services Description Language) ที่บอกถึงไทป์ของข้อมูลที่รับหรือส่ง หรือใช้ xsi:type แอตทริบิวต์ในกรณีที่ภาษาที่ใช้ไม่รองรับ WSDL โดยทั้ง 2 วิธีจะใช้โครงสร้างของ XML ในการระบุไทป์ ตัวอย่างของการใช้ xsi จะเป็นดังนี้

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
        soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <soap:Body>
    <m:MixedMessage xmlns:m="http://www.wrox.com/mix/">
      <param1 xsi:type="xsd:string">OU812</param1>
      <param2 xsi:type="xsd:integer">2001</param2>
      <param3 xsi:type="xsd:double">3.14159</param3>
    </m:MixedMessage>
  </soap:Body>
</soap:Envelope>
```

### 3.1.5 SOAP ใน HTTP

ในการส่ง SOAP ผ่านทาง HTTP จะต้องใช้ content-type เป็น text/xml แต่ใน SOAP ร้องขอจะต้องมีเฮดเดอร์ SOAP Action ภายใน HTTP เฮดเดอร์ SOAP Action จะเป็นตัวบอกให้เซิร์ฟเวอร์รู้ว่า HTTP Post นั้นเป็น SOAP เมสเซจ และค่าของ เฮดเดอร์ คือ URI ที่แสดงถึงจุดหมายของ SOAP เมสเซจ ส่วน SOAP ตอบสนองจะต้องมี status code ตามมาตรฐานของ HTTP โดย 200-299 แสดงว่าสำเร็จ แต่ถ้า ตอบสนองเมสเซจ เป็นการฟอลต์ แล้ว status code จะต้องเป็น 500 ซึ่งแสดงถึง internal server error ตัวอย่างของตอบสนองที่มี status code เป็น 500 อาจเป็นดังนี้

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/xml
Content-Length: ###
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
        soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <soap:Body>
    <soap:Fault>
      <faultcode>soap:VersionMismatch</faultcode>
        <faultstring>The SOAP เนมสเปซ is incorrect.</faultstring>
      <faultactor>http://www.wrox.com/endpoint.asp</faultactor>
      <detail>
        <w:errorinfo xmlns:w="http://www.wrox.com/">
          <desc>The SOAP เนมสเปซ was blank.</desc>
        </w:errorinfo>
      </detail>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

### 3.1.6 SOAP สำหรับ RPC

จุดประสงค์ของการออกแบบ SOAP คือทำ RPC โดยใช้ XML การเรียก RPC นั้นจะ map เข้ากับ HTTP ร้องขอ และ RPC ตอบสนองจะ map กับ HTTP ตอบสนอง

ในการทำ mechod call จำเป็นที่จะต้องมีอินฟอร์เมชันดังนี้

1. URI ของออบเจ็กต์ปลายทาง (target object)
2. ชื่อของเมธอด
3. คำอธิบายของเมธอด (method signature) ส่วนนี้เป็นอ็อปชัน
4. พารามิเตอร์สำหรับเมธอด
5. ข้อมูลส่วนหัว (header data) เป็นอ็อปชัน

### 3.1.7 SOAP Toolkit

SOAP toolkit คือเครื่องมือที่จะทำหน้าที่ในการประมวลผลการร้องขอ SOAP และส่งการตอบสนองกลับไปให้ไคลเอนต์ โดย SOAP toolkit แต่ละตัวใช้ได้บนแพลตฟอร์มที่ต่างกัน รองรับเทคโนโลยีเว็บเซอร์วิสต่างกัน บางตัวอาจจะรองรับ WSDL หรือ UDDI ในขณะที่บางตัวรองรับอย่างใดอย่างหนึ่งหรือไม่รองรับทั้งสองอย่าง และถึงแม้ว่า SOAP จะเป็นอิสระต่อแพลตฟอร์ม แต่ใน SOAP toolkit บางตัวยังไม่สามารถทำงานข้ามผลิตภัณฑ์ (interoperability) ได้ เนื่องจากรองรับเทคโนโลยีของเว็บเซอร์วิส ไม่เหมือนกัน เช่น xsi ในบางผลิตภัณฑ์จะบังคับให้ SOAP เมสเซจจะต้องระบุไทป์ด้วย xsi หากไม่มีก็จะความผิดพลาดซึ่งในปัจจุบันแต่ละผลิตภัณฑ์ก็พยายามพัฒนาให้สามารถทำงานด้วยกันได้

## 3.2 UDDI

UDDI (Universal Description, Discovery and Integration) เป็นมาตรฐานที่จัดตั้งขึ้น โดยบริษัทไอบีเอ็ม, ไมโครซอฟท์และบริษัทยักษ์ใหญ่ทางธุรกิจ B2B (Business-to-Business) อื่น ๆ UDDI ถูกสร้างขึ้นมาเป็นมาตรฐาน ในการค้นหาบริการเว็บเซอร์วิสสำหรับคู่ค้าทางธุรกิจ ซึ่งเปรียบได้กับฐานข้อมูล ขนาดใหญ่ ซึ่งมีข้อมูลของเว็บเซอร์วิสที่เปิดให้บริการ โดยที่เว็บไซต์สำหรับค้นหาเว็บเซอร์วิสมีอยู่หลายที่อย่างเช่น

- http://uddi.microsoft.com/search.aspx
- http://www-3.ibm.com/services/uddi/testregistry/find
- http://uddi.org

## 2.4 WSDL

WSDL (Web Services Description Language) เกิดจากความร่วมมือระหว่างบริษัทไอบีเอ็มและไมโครซอฟท์ WSDL เป็นภาษาที่ใช้อธิบายคุณลักษณะของเว็บเซอร์วิสและวิธีการติดต่อกับเว็บเซอร์วิสนั้น ๆ โดยใช้ไวยากรณ์ของภาษา XML ซึ่ง WSDL อยู่ในความดูแลของ W3C รายละเอียด สามารถหาอ่านเพิ่มเติมได้จากเว็บไซต์ของ W3C ที่ http://www.w3.org/TR/wsdl ส่วนในทางปฏิบัติ หากเราต้องการ สร้างเว็บเซอร์วิสขึ้นมาเป็นของตนเอง ก็สามารถสร้างเอกสาร WSDL ได้โดยอัตโนมัติ เราจึงไม่ต้องไปกังวลในรายละเอียด ในข้อกำหนดใน WSDL มากนักโดยตัวอย่างเอกสาร WSDL เช่น

```
<?xml version="1.0" encoding="utf-8" ?>
- <definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:conv="http://www.openuri.org/2002/04/soap/conversation/"
    xmlns:cw="http://www.openuri.org/2002/04/wsdl/conversation/"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:jms="http://www.openuri.org/2002/04/wsdl/jms/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    xmlns:s="http://www.w3.org/2001/XMLSchema"
    xmlns:s0="http://www.openuri.org/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    targetNamespace="http://www.openuri.org/">
- <types>
- <s:schema xmlns:s="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified" targetNamespace="http://www.openuri.org/">
- <s:element name="plus">
- <s:complexType>
- <s:sequence>
  <s:element name="i" type="s:int" />
  <s:element name="j" type="s:int" />
    </s:sequence>
    </s:complexType>
    </s:element>
- <s:element name="plusResponse">
- <s:complexType>
- <s:sequence>
  <s:element name="plusResult" type="s:int" />
    </s:sequence>
    </s:complexType>
    </s:element>
  <s:element name="int" type="s:int" />
    </s:schema>
    </types>
- <message name="plusSoapIn">
  <part name="parameters" element="s0:plus" />
    </message>
- <message name="plusSoapOut">
  <part name="parameters" element="s0:plusResponse" />
    </message>
- <message name="plusHttpGetIn">
  <part name="i" type="s:string" />
  <part name="j" type="s:string" />
    </message>
- <message name="plusHttpGetOut">
  <part name="Body" element="s0:int" />
    </message>
- <message name="plusHttpPostIn">
  <part name="i" type="s:string" />
  <part name="j" type="s:string" />
    </message>
- <message name="plusHttpPostOut">
  <part name="Body" element="s0:int" />
    </message>
- <portType name="serviceSoap">
```

```
- <operation name="plus">
  <input message="s0:plusSoapIn" />
  <output message="s0:plusSoapOut" />
    </operation>
    </portType>
- <portType name="serviceHttpGet">
- <operation name="plus">
  <input message="s0:plusHttpGetIn" />
  <output message="s0:plusHttpGetOut" />
    </operation>
    </portType>
- <portType name="serviceHttpPost">
- <operation name="plus">
  <input message="s0:plusHttpPostIn" />
  <output message="s0:plusHttpPostOut" />
    </operation>
    </portType>
- <binding name="serviceSoap" type="s0:serviceSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
    style="document" />
- <operation name="plus">
  <soap:operation soapAction="http://www.openuri.org/plus" style="document" />
- <input>
  <soap:body use="literal" />
    </input>
- <output>
  <soap:body use="literal" />
    </output>
    </operation>
    </binding>
- <binding name="serviceHttpGet" type="s0:serviceHttpGet">
  <http:binding verb="GET" />
- <operation name="plus">
  <http:operation location="/plus" />
- <input>
  <http:urlEncoded />
    </input>
- <output>
  <mime:mimeXml part="Body" />
    </output>
    </operation>
    </binding>
- <binding name="serviceHttpPost" type="s0:serviceHttpPost">
  <http:binding verb="POST" />
- <operation name="plus">
  <http:operation location="/plus" />
- <input>
  <mime:content type="application/x-www-form-urlencoded" />
    </input>
- <output>
  <mime:mimeXml part="Body" />
    </output>
    </operation>
```

```
    </binding>
- <service name="service">
- <port name="serviceSoap" binding="s0:serviceSoap">
  <soap:address location="http://localhost:7001/testWeb/service/service.jws" />
    </port>
- <port name="serviceHttpGet" binding="s0:serviceHttpGet">
  <http:address location="http://localhost:7001/testWeb/service/service.jws" />
    </port>
- <port name="serviceHttpPost" binding="s0:serviceHttpPost">
  <http:address location="http://localhost:7001/testWeb/service/service.jws" />
    </port>
    </service>
    </definitions>
```

# บทที่ 4

# Security

Security หมายถึง เทคนิคในการที่จะมั่นใจได้ว่าข้อมูลที่ถูกเก็บอยู่ในคอมพิวเตอร์ หรือข้อมูลที่ถูกส่งผ่านระหว่างเครื่องคอมพิวเตอร์หรือเครือข่ายคอมพิวเตอร์จะไม่ถูกเปลี่ยนแปลง โดยหลักการวัด security โดยทั่วไปนั้นส่วนใหญ่จะเกี่ยวกับการ พิสูจน์ข้อความ และการเข้ารหัส

การส่งข้อมูลในเครือข่ายนั้นจะมี accesspoint หลายจุดซึ่งผู้ประสงค์ร้ายจะสามารถดักจับ และเปลี่ยนแปลงข้อมูลได้ โดยเฉพาะยิ่งธุรกิจมีการกระจายตัวมากเท่าไรก็จะยิ่งเพิ่มโอกาสที่จะโดนโจมตีได้มากขึ้นจึงมีความต้องการ software ที่จะมาช่วยจัดการในด้านนี้

## 4.1 Cipher Suite

Cipher Suite คือ กระบวนการ encryption ของ SSL ที่รวม key exchange algorithm,symmetric encryption และ secure hash algorithm เข้าด้วยกัน ตัวอย่างเช่น Cipher Suite ที่เรียกว่า RSA_WITH_RC4_128MD5 ใช้ RSA สำหรับ key exchange , RC4 ที่ใช้ 128-bit key ในการ encryption และ MD5 สำหรับการทำ Message digest

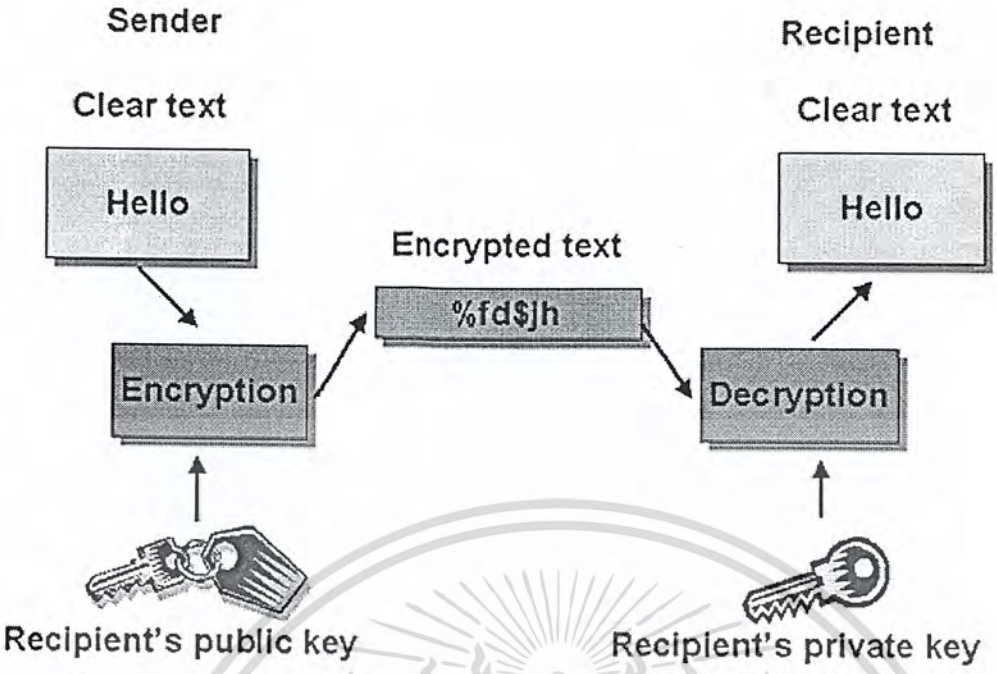
## 4.2 Public Key Algorithms

Public key(or asymmetric key)algorithms เป็นการใช้ key 2 key ที่แตกต่างกัน แต่มีความสัมพันธ์กันทางคณิตศาสตร์

- A public key(ที่ซึ่งถูกแจกจ่ายโดยทั่วไป) ใช้สำหรับการยืนยัน digital signatureว่าเป็นของผู้ส่งจริง หรือ ใช้ในการเปลี่ยนข้อมูลให้อยู่ในรูปที่ไม่สามารถเข้าใจได้
- A private key(ที่ซึ่งถูกเก็บเป็นความลับ) ใช้ในการสร้าง digital signature หรือ เปลี่ยนข้อมูลที่ถูกส่งมาให้กลับอยู่ในรูปเดิม

## 4.3 Symmetric Key Algorithms

ใน symmetric key algorithms จะใช้ key เดียวกันทั้ง encrypt และ decrypt message โดยการใช้วิธีนี้จะมีความรวดเร็วกว่าการทำแบบ public key cryptography มาก แต่มีข้อเสียคือปัญหาการแลกเปลี่ยน key




*รูปที่ 4-1 Public Key*

**4.4 Message Digest Algorithms**

สนับสนุน message digest algorithm แบบ MD5 และ SHA(Secure Hash Algorithm) ซึ่งเป็น one-way hash algorithm โดย one-way hash algorithm จะ convert message ให้อยู่ในรูปแบบ string ที่ fix length ซึ่งเรียกว่า message digest หรือ hash value

MD5 จะเป็น high-speed 128-bit hash ส่วน SHA จะมีความปลอดภัยมากกว่าโดยใช้ 160-bit hash แต่จะช้ากว่า MD5

**4.5 ลายมือชื่อดิจิตอล**

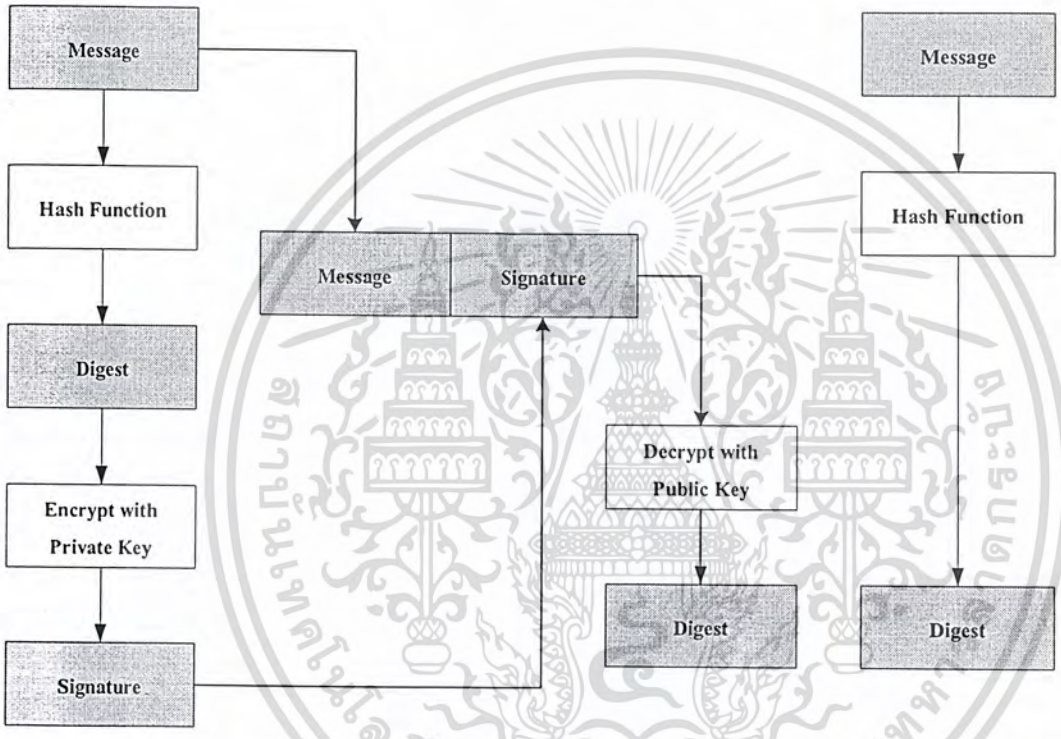
ลายมือชื่อดิจิตอลใช้เมื่อต้องการความมั่นใจในแหล่งที่มาของเอกสาร เปรียบเสมือนลายมือชื่อซึ่งเฉพาะเจ้าของจริงที่สามารถคำนวณขึ้นได้ แต่ลายมือชื่อนี้สามารถพิสูจน์ได้ กล่าวคือบุคคลอื่นสามารถตรวจสอบได้ว่าลายมือชื่อนั้นมาจากผู้สร้างจริงๆ วิธีที่ใช้ในการทำลายมือชื่อดิจิตอลที่มีความปลอดภัยค่อนข้างสูงก็คือ การใช้ public-private key โดยเข้ารหัสค่าลายมือชื่อด้วยคีย์ส่วนตัว และคนอื่นสามารถใช้คีย์สาธารณะพิสูจน์ได้ว่าลายมือชื่อมาจากคีย์ส่วนตัวที่ตรงกัน

คุณสมบัติที่สำคัญของลายมือชื่อดิจิตอลที่สำคัญนั้นจะต้องประกอบด้วย 2 ประการคือ

- สามารถยืนยันได้ว่าข้อมูลที่รับมานั้นไม่มีการเปลี่ยนแปลงระหว่างการส่ง
- สามารถยืนยันได้ว่าข้อมูลนั้นได้รับการยืนยันจากผู้ลงลายมือชื่อจริงๆ

การทำลายมือชื่อดิจิตอลนั้นทำได้โดยนำข้อความแรกเริ่มไปเข้าแฮชฟังก์ชันจะได้เป็นเมสเซจไดเจสต์ เพื่อเป็นการยืนยันความถูกต้องของข้อมูลก่อน จากนั้นจะนำเมสเซจไดเจสต์ที่ได้ส่งไปเข้ารหัสกับคีย์ส่วนตัวของผู้ส่งเพื่อเป็นการยืนยันว่าเป็นเจ้าของเอกสารนั้นจริง และจะได้สิ่งที่เรียกว่าลายมือชื่อดิจิตอลออกมาส่งไปพร้อมกับข้อมูลด้วย ทางด้านรับนั้นจะเป็นวิธีการที่คล้ายกับด้านส่งโดยจะต้องเริ่มจากการหาเมสเซจไดเจสต์ของข้อมูลที่ได้

รับมาเก็บไว้ก่อน จากนั้นจะนำลายมือชื่อดิจิตอลที่ได้รับมาด้วย มาทำการถอดรหัสด้วยคีย์สาธารณะของผู้ส่ง ในขั้นตอนนี้เองที่จะสามารถยืนยันบุคคลผู้ส่งได้ เพราะถ้าหากผู้ที่ลงลายมือชื่อไม่ได้เป็นบุคคลที่เราคาดว่าเป็นเจ้าของก็จะไม่สามารถถอดลายมือดิจิตอลได้ ถ้าหากการยืนยันบุคคลผู้ส่งสำเร็จแล้วจะได้รับเมสเซจไดเจสต์ของข้อมูลก่อนจะส่งออกมา ด้านรับจะต้องทำการตรวจสอบความถูกต้องของข้อมูลโดยการนำเมสเซจไดเจสต์ที่คำนวณได้ในตอนรับข้อมูล กับเมสเซจไดเจสต์ที่ได้หลังจากการถอดรหัสลายมือชื่อดิจิตอลออกมา นำมาเปรียบเทียบกัน ถ้าหากเมสเซจไดเจสต์ที่ได้ออกมามีค่าเท่ากันนั่นแสดงว่าข้อมูลที่ได้รับนั้นเป็นข้อมูลเดียวกันจากทางด้านส่ง และไม่มีการเปลี่ยนแปลงข้อมูลระหว่างการส่ง ขั้นตอนการส่ง และการรับของวิธีการนี้



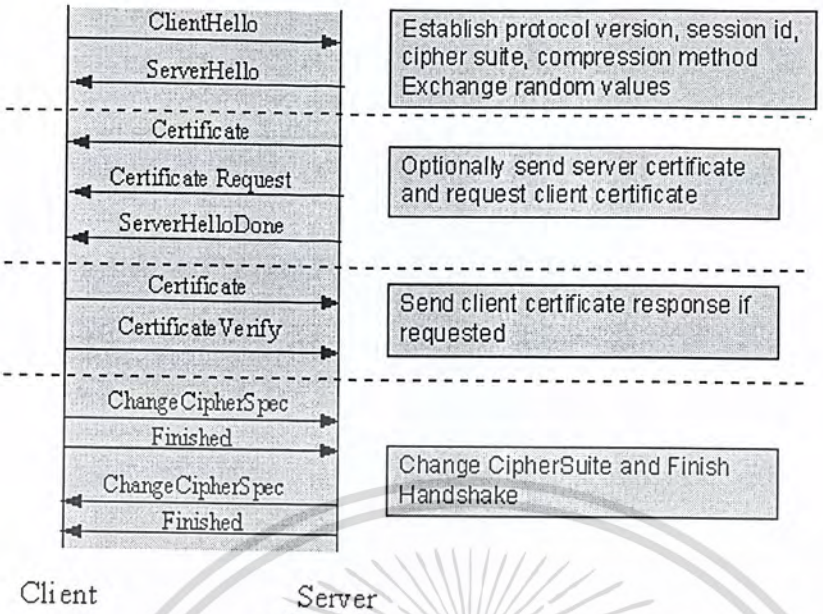

*รูปที่ 4-2 Digital Signature*

**4.6 SSL Protocol**

SSL Protocol สร้างความปลอดภัยให้กับ application ที่ทำการติดต่อผ่าน network ซึ่งมีรายละเอียดดังนี้

- กลไกในการที่แต่ละ application สามารถใช้ยืนยันหลักฐานของ application อื่นได้
- เข้ารหัสข้อมูลที่ใช้ในการแลกเปลี่ยนระหว่าง application

เมื่อ SSL Protocol ถูกใช้ target จะทำการยืนยันตัวเองกับ initiator ถ้า target request initiator จะสามารถยืนยันตัวเองกับ target ได้ SSL connection จะเริ่มด้วยการ handshake ระหว่าง application จะ แลกเปลี่ยน didital certificate กัน ตกลง encryption algorithm ที่จะใช้ และ สร้าง encryption key ที่จะใช้ตลอดช่วงการติดต่อ



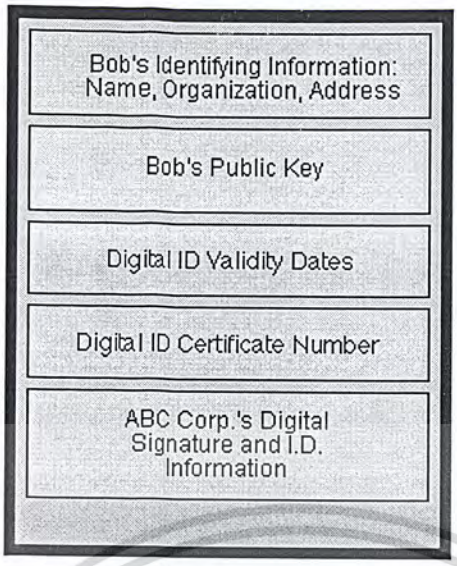

*รูปที่ 4-3 SSL-Handshake*

**4.7 เอกสารสิทธิ์(Digital Certificates)**

ลักษณะของเอกสารสิทธิ์ดิจิตอล

เอกสารสิทธิ์เป็นเสมือนกับบัตรประจำตัวประชาชน ซึ่งจะบ่งบอกรายละเอียดของบุคคล เพื่อใช่ในการยืนยันตัวตน เราสามารถส่งเอกสารสิทธิ์ไปควบคู่กับลายมือชื่อดิจิตอลเพื่อใช่ในการยืนยันว่าเอกสารสิทธิ์นั้นไม่มีการเปลี่ยนแปลง การสร้างเอกสารสิทธิ์ทำโดยผู้ใช้ทุกคนทำการขอเอกสารสิทธิ์กับองค์กรพิสูจน์สิทธิ์ (Certificate Authority : CA) โดยการส่งคีย์สาธารณะและข้อมูลตามที่องค์กรพิสูจน์สิทธิ์นั้นกำหนดไปและทำการขอเอกสารสิทธิ์มา การติดต่อต้องทำเป็นการส่วนตัวหรือติดต่อผ่านระบบการพิสูจน์ตัวตนที่ปลอดภัย เราสามารถกำหนดสิ่งที่จำเป็นในการสื่อสารได้ดังต่อไปนี้

1. ผู้ใช้ทุกคนสามารถคำนวณหาชื่อและคีย์สาธารณะของเจ้าของเอกสารสิทธิ์ได้
2. ผู้ใช้ทุกคนสามารถตรวจสอบได้ว่าเอกสารสิทธิ์มาจากองค์กรพิสูจน์สิทธิ์จริงๆไม่ได้ถูกปลอมแปลงมา
3. ผู้ใช้สามารถตรวจสอบได้ว่าเอกสารสิทธิ์นั้นหมดอายุหรือไม่
4. ผู้ที่สามารถอ้างและอัพเดต(update)เอกสารสิทธิ์ได้มีเพียงองค์กรมีอำนาจในการรับรองสิทธิ์เท่านั้น
5. ผู้ใช้ทุกคนสามารถตรวจสอบเอกสารสิทธิ์ได้เป็นประจำ




*รูปที่ 4-4 Certificate*

**Certificate Authority**

Digital certificate จะถูกออกโดย certificate authority ที่ซึ่งเป็นองค์กร(บุคคลที่สาม)ที่เชื่อถือได้ เมื่อ certificate authority สร้าง digital certificate แล้ว จะทำการเข้ารหัสด้วย private key ของcertificate authority นั้น แล้วจะทำการส่งไปให้กับผู้ที่ขอเอกสารสิทธ์นั้น

โดยผู้ขอเอกสารสิทธิ์จะยืน ยัน ตัวของ certificate authority นั้น ด้วย public key ซึ่งถูกออกด้วย high-level certificate authority ซึ่งมีความน่าเชื่อถือกว่า

**บริการพิสูจน์สิทธิ์แบบ X.509 (X.509 Authentication Service)**

ระบบ X.509 เป็นระบบพิสูจน์สิทธ์ที่สำคัญมากในระบบเครือข่าย โดย X.509 เป็นอนุกรมย่อย ของ X.500 ซึ่งกำหนดมาตรฐาน ITU-T โดยขณะที่ X.500 เป็นตัวกำหนดโครงสร้างในลักษณะที่เป็นไดเรกทอรี หรือโครงสร้างนั้น X.509 จะทำหน้าที่ในการพิสูจน์สิทธิ์ให้กับส่วนต่าง ๆ ของไดเรกทอรีนั้น สำหรับรูปแบบการใช้งานจะเน้นไปที่การพิสูจน์บุคคล เพื่อยืนยันการติดต่อเป็นสำคัญ

การทำงานของ X.509 จะมีโครงสร้างการทำงานที่เป็นไดเรกทอรี โดยในที่นี้ไดเรกทอรี จะทำหน้าที่เป็นที่เก็บข้อมูลที่ใช้ในการยืนยัน ซึ่งโดยทั่วไปจะอยู่ในรูปของเอกสารสิทธิ์ซึ่งในเอกสารสิทธิ์จะบรรจุคีย์สาธารณะของผู้ใช้ที่เข้ารหัสโดยคีย์ส่วนตัวขององค์กรที่จ่ายใบเอกสารสิทธิ์มาให้ สำหรับการทำงานของ X.509 นั้นจะมีขอบเขตการนำไปใช้งานที่กว้างขวางมาก เช่น ใช้ในการทำ Mail Security ใช้ในการทำ IP Security ใช้ในการทำ Web Security หรือหากจะกล่าวว่า เมื่อใดที่ต้องการพิสูจน์บุคคลหรือยืนยันเครื่องคอมพิวเตอร์แล้ว ก็มักจะอยู่ในขอบข่ายการทำงานของ X.509 เสมอ

X.509 ได้ถูกนำเสนอเมื่อปี 1988 จากนั้นได้ผ่านการปรับปรุงเป็นลำดับขั้น ในประเด็น

ต่าง ๆ รวมทั้งเรื่องความปลอดภัยด้วย จากนั้นก็ได้ออกมาเป็นข้อเสนอที่ปรับปรุงแล้วในปี 1993 และปรับปรุงอีกครั้งในปี 1995 โดยการทำงานของ X.509 จะใช้การเข้ารหัสแบบคีย์สาธารณะแล้วใช้มาตรฐานลายมือชื่อดิจิตอลในการรับรองข้อมูล สำหรับอัลกอริธึมนั้น ไม่ได้ระบุแน่นอนโดยสามารถเลือกใช้ได้หลายตัวแต่ที่แนะนำคืออาร์เอสเอ

รูปแบบทั่วไปของเอกสารสิทธิ์

- เวอร์ชัน (Version) แสดงหมายเลขเวอร์ชัน เพราะในแต่ละเวอร์ชันจะมีรูปแบบของข้อมูลที่ไม่เหมือนกันก็ได้ โดยปกติจะเป็นเวอร์ชัน 1 แต่หากในเอกสารสิทธิ์มีการใช้
- หมายเลขลำดับ (Serial Number) เป็นเลขจำนวนเต็ม โดยจะต้องไม่ซ้ำกันในองค์กรที่ออกเอกสารสิทธิ์ โดยเลขนี้จะเป็นเลขที่จะใช้อ้างถึงแต่ละเอกสารสิทธิ์ที่สร้างขึ้นมา
- อัลกอริธึมที่ใช้สร้าง (Signature Algorithm Identifier) เป็นฟิลด์ที่ระบุอัลกอริธึมที่ใช้ในการสร้างเอกสารสิทธิ์
- ชื่อผู้ออกเอกสารสิทธิ์ (Issue Name) เป็นชื่อขององค์กรที่ออกเอกสารสิทธิ์
- ช่วงเวลาที่รับรองเอกสารสิทธิ์ (Period of Validity) เป็นตัวบอกว่า ให้ใช้เอกสารสิทธิ์นี้ตั้งแต่ วันที่เท่าไหร่และสิ้นสุดวันที่เท่าไหร่
- ชื่อเจ้าของเอกสารสิทธิ์ (Subject Name) เป็นชื่อของบุคคลที่เอกสารสิทธิ์ใบนี้อ้างถึงหรือ แทนตัวบุคคลนั้น
- ข้อมูลของคีย์สาธารณะ (Subject's Public Key Information) เป็นฟิลด์ที่เก็บคีย์สาธารณะและระบุถึงอัลกอริธึมที่ใช้กับคีย์นี้ขึ้นมา รวมถึงพารามิเตอร์อื่น ๆ ด้วย
- ตัวระบุผู้ออกเอกสารสิทธิ์ (Issuer Unique Identifier) เป็นฟิลด์ออปชันที่ใช้ในการระบุถึงองค์กรที่ออกเกอสารสิทธิ์ ในกรณีที่ชื่อ X.500 มีการนำไปใช้กับส่วนอื่น ๆ
- ตัวระบุผู้ขอเอกสารสิทธิ์ (Subject Unique Identifier) เป็นฟิลด์ที่ใช้ในการระบุถึงตัวบุคคลที่เป็นเจ้าของเอกสารสิทธิ์ ในกรณีที่ชื่อ X.500 มีการนำไปใช้กับส่วนอื่น ๆ
- ส่วนขยาย (Extension) เป็นกลุ่มของฟิลด์ที่เพิ่มเติมข้อมูลอื่น ๆ เข้ามาด้วย
- ลายมือชื่อ (Signature) จะบรรจุเมสเซจไดเจสต์ของข้อมูลในทุกฟิลด์ที่เข้ารหัสด้วยคีย์ส่วนตัวขององค์กรพิสูจน์สิทธิ์เพื่อเป็นการยืนยันว่า เอกสารสิทธิ์นี้สร้างมาจากองค์กรดังกล่าวจริง ๆ โดยจะมีข้อมูลที่ระบุวิธีการแฮชและวิธีการเข้ารหัสด้วย

ในการใช้งานเอกสารสิทธิ์จะมีช่วงเวลาใช้งานที่จำกัดแน่นอน ดังนั้นหากผู้ใช้ต้องการใช้เอกสารสิทธ์ต่อไป ก็ต้องขอต่ออายุเอกสารสิทธิ์ก่อนที่จะหมดอายุ แต่หากจะมีการหมดอายุโดยที่ไม่ขอต่อหรือมีการเลิกใช้เอกสารสิทธิ์อาจจะเนื่องมาจากพนักงานลาออก หรืออาจจะเนื่องจากเอกสารสิทธินี้ไม่ปลอดภัยแล้วก็ต้องทำการเรียกคืน (Revoke) และองค์กรพิสูจน์สิทธิ์จะต้องมีการจัดทำรายการเอกสารสิทธิ์ที่ถูกเรียกคืน (Certificate Revocation List - CRL) ซึ่งจะเก็บไว้ในไดเรกทอรีและรับรองโดยองค์กรพิสูจน์สิทธิ์ซึ่งผู้ที่ต้องการตรวจสอบเอกสารสิทธิ์ว่าเป็นเอกสารสิทธิ์ที่ไม่ใช้งานแล้วหรือไม่ ก็ต้องขอรายการเอกสารสิทธิ์ที่ถูกเรียกคืนไปตรวจสอบ

และเนื่องจากเอกสารสิทธิ์ไม่สามารถปลอมได้ ดังนั้นการเก็บเอกสารสิทธิ์ไว้ที่องค์กร

พิสูจน์สิทธิ์จึงไม่ต้องมีกลไลพิเศษมาป้องกันแต่อย่างใด กล่าวคือผู้ใช้คนใดที่เป็นสมาชิกขององค์กรพิสูจน์สิทธิ์ก็สามารถเข้าถึงเอกสารสิทธ์ของผู้ใช้คนอื่น ๆ ได้ทุกคน โดยเอกสารสิทธิจะเก็บอยู่ในไฟล์เพียงไฟล์เดียวที่มีขนาดเล็ก นอกจากจะสามารถขอเอกสารสิทธ์จากองค์กรพิสูจน์สิทธิ์แล้วผู้ใช้ยังสามารถส่งเอกสารสิทธ์ไปให้กันเองได้อีกด้วย โดยผ่านทางสื่อต่าง ๆ เช่น จดหมายอิเล็กทรอนิกส์, ส่งผ่านแผ่นดิสก์เก็ต เป็นต้น

อย่างไรก็ตาม เนื่องจากระบบเครือข่ายในปัจจุบันมีขนาดใหญ่โตกว้างขวางมาก และการติดต่อสื่อสารก็ไม่ได้มีลักษณะเฉพาะกลุ่มอีกแล้ว ดังนั้นการที่จะให้ผู้ใช้ทุกคนมาใช้เอกสารสิทธ์ที่รับรองโดยองค์กรพิสูจน์สิทธิ์เดียวกันทั้งหมดก็อาจเป็นเรื่องยาก หากผู้ใช้ 2 คนใช้เอกสารสิทธิ์ที่รับรองจากองค์กรพิสูจน์สิทธิ์คนละแห่งก็จะไม่สามารถตรวจสอบเอกสารสิทธิ์ของอีกฝ่ายได้ว่าเป็นฉบับจริงหรือไม่ ซึ่งในกรณีเช่นนี้ ก็อาจจะใช้วิธีสำเนาคีย์สาธารณะขององค์กรพิสูจน์สิทธิ์ของผู้ใช้อีกคนหนึ่งมาทำการตรวจสอบเองก็สามารถทำได้ เช่น กำหนดให้มี CA-A และ CA-B โดยให้บริการกับผู้ใช้ A และ B แต่เนื่องจากในครั้งแรกที่ A สำเนาคีย์สาธารณะของ CA-B มานั้นอาจเกิดการปลอมแปลงได้ เพราะสิ่งที่เรามีอยู่ก็คือคีย์สาธารณะของ CA-A ของเรา แต่เอกสารสิทธิ์ของ CA-B ทำให้เราไม่สามารถตรวจสอบว่าเอกสารสิทธิ์ที่ได้รับมานั้นเป็นฉบับที่ถูกต้องหรือไม่ ดังนั้นวิธีดังกล่าวจึงถือว่า ไม่มีความปลอดภัยเพียงพอ

สำหรับอีกวิธีการอีกแบบ คือให้ CA-A เก็บเอกสารสิทธิ์ของ CA-B เอาไว้ด้วยและ CA-B ก็เก็บเอกสารสิทธิ์ของ CA-A เอาไว้เช่นกัน ด้วยวิธีนี้เราก็สามารถให้องค์กรพิสูจน์สิทธิ์ตรวจสอบเอกสารสิทธิ์ได้ไม่ว่าเอกสารสิทธิ์นั้นจะรับรองจาก CA-A หรือ CA-B ก็ตาม เช่น ผู้ใช้ A ต้องการตรวจสอบเอกสารสิทธิ์ที่รับรองจาก CA-A ก็สามารถทำได้เลยเพราะรู้คีย์สาธารณะของ CA-A อยู่แล้วเนื่องจากเป็นสมาชิก CA-A และหากผู้ใช้ A ต้องการตรวจสอบเอกสารสิทธิ์ที่รับรองโดย CA-B ผู้ใช้ A ก็ขอเอกสารสิทธิ์ของ CA-B จาก CA-A โดยเอกสารสิทธิ์ดังกล่าวจะรับรองโดย CA-A ดังนั้นจึงแน่ใจได้ว่าเอกสารสิทธิ์ของ CA-B ที่ได้รับนั้นเป็นของจริงและคีย์สาธารณะของ CA-B ก็เป็นของจริง

### 4.8 WS-Security

คือขบวนการ security ในระดับ message base security ในระดับ xml เพื่อยืนยัน message integrity, message confidentiality และ message authentication โดย WS-Security ทำโดยการเพิ่ม Security Token เข้ากับ Message โดย Security Token ไม่มีการกำหนดรูปแบบแน่นอน ซึ่งจุดประสงค์ของ WS-Security คือการให้แอพพลิเคชันสามารถสร้าง Secure Soap message ในรูปแบบ end-to-end message-level security ระหว่างกันได้

#### 4.8.1 Token

WS-Security แบ่ง Security Token ออกเป็น 2 รูปแบบคือ

1 Unsigned Security Token เช่น Username Token

2 Signed Security Token เช่น X.509 Certificate, Kerberos Ticket

ตัวอย่างการเซท WS-Policy ให้เว็บเซอร์วิสใช้ Username Token

```
<S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-envelope"
        xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext">
   <S:Header>
        ...
      <wsse:Security>
        <wsse:UsernameToken>
          <wsse:Username>WebLogic</wsse:Username>
          <wsse:Password>WebLogic</wsse:Password>
        </wsse:UsernameToken>
      </wsse:Security>
        ...
   </S:Header>
   ...
</S:Envelope>
```

**4.8.2 XML-Signature**

จุดประสงค์ของ XML-Signature คือการยืนยันตัวตนของผู้ส่งเพื่อพิสูจน์ว่าเอกสารที่ส่งมาไม่ถูกเปลี่ยนแปลง ซึ่งถ้าหากเอกสารถูกเปลี่ยนแปลงระหว่างการส่งเมื่อทำการตรวจสอบจะได้ผลไม่ตรงกันทำให้บอกได้ว่าเอกสารนี้ส่งถูกส่งจากผู้ส่งที่เราคาดหวังจริงหรือไม่ ซึ่ง WS-Security สามารถทำการ XML-Signature ได้หลายส่วนของเอกสาร หรือ บางส่วนของเอกสารได้

ตัวอย่างการเซท WS-Policy ให้เว็บเซอร์วิสใช้ XML-Signature

```
<wsSecurityPolicy xsi:schemaLocation="WSSecurity-
policy.xsd"xmlns="http://www.bea.com/2003/03/wsse/config"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <wsSecurityIn>
      <!--
      Incoming SOAP messages must be digitally signed with the sender's
      private key.
      The sender's public key is used to validate the signature.
      -->
                <signatureRequired>true</signatureRequired>
        </wsSecurityIn>
  <wsSecurityOut>
      <!--
      Sign the SOAP message with the sender's private key. Only the sender's public
 key can validate the signature. Ensures the authenticity of the sender, i.e., that the sender is
      in fact the source of the SOAP message.
      -->
      <signatureKey>
                <alias>mycompany</alias>
                <password>password</password>
        </signatureKey>
  </wsSecurityOut>
  <!--
Look for the sender's.jks keystore in the default location, the server domain root, in this case,
BEA_HOME\WebLogic81\samples\domains\workshop.
```

```
-->
<keyStore>
                <keyStoreLocation>samples_sender.jks</keyStoreLocation>
                <keyStorePassword>password</keyStorePassword>
        </keyStore>
</wsSecurityPolicy>
```

**4.8.3 XML-Encryption**

จุดประสงค์การทำ XML-Encryption คือเพื่อให้บุคคลที่เราไม่ต้องการให้รู้รายละเอียดของเอกสารล่วงรู้รายละเอียดซึ่งเป็นการยืนยัน Privacy ของเอกสาร ซึ่ง WS-Security สามารถทำการ XML-Encryption เฉพาะ element บางตัวของเอกสาร XML ได้

ตัวอย่างการเซท WS-Policy ให้เว็บเซอร์วิสใช้ XML-Encryption

```
<wsSecurityPolicy xsi:schemaLocation="WSSecurity-policy.xsd"
xmlns="http://www.bea.com/2003/03/wsse/config"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <wsSecurityOut>
    <!--
Encrypt the SOAP message with the recipient's public key. Only the recipient's private key can decrypt
it.
Ensures the confidentiality of the SOAP message. (This process requires that the sender's keystore
already
contain a digital certificate containing the recipient's public key.)
    -->
          <encryption>
                <encryptionKey>
                        <alias>mycompany</alias>
                </encryptionKey>
          </encryption>
        </wsSecurityOut>

        <wsSecurityIn>
            <!--Incoming SOAP messages must be encrypted with client's public key. The alias and
            password to access the client's decrypting private key in the keystore are provided by the
              <decryptionKey> element below.      -->
                <encryptionRequired>
                        <decryptionKey>
                                <alias>client1</alias>
                                <password>password</password>
                        </decryptionKey>
                </encryptionRequired>
</wsSecurityIn>

<!--
Look for the client.jks keystore in the default location, the server domain
root, in this case, BEA_HOME\WebLogic81\samples\domains\workshop.
-->
        <keyStore>
 <keyStoreLocation>samples_client.jks</keyStoreLocation>
```

```
        <keyStorePassword>password</keyStorePassword>
    <</keyStore>
</wsSecurityPolicy>
```

# บทที่5

# ผลการทดลอง

## 5.1 Xml – Encryption in Soap

**Service Request Plus**

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Header>

    <wsse:Security env:mustUnderstand="1" xmlns:wsse="http://docs.oasis-
      open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
      <xenc:EncryptionMethod
        Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-
        1_5"></xenc:EncryptionMethod>
        <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
          <wsse:SecurityTokenReference>
            <dsig:X509IssuerSerial>
            <dsig:X509IssuerName>CN=MyCompany, OU=Development,
              O=MyDevTeam, L=Sealand, ST=WA,
              C=US</dsig:X509IssuerName>
              <dsig:X509SerialNumber>1051745810</dsig:X509SerialNumber>
            </dsig:X509IssuerSerial>
          </wsse:SecurityTokenReference>
        </dsig:KeyInfo>
          <xenc:CipherData>
          <xenc:CipherValue>BgFMb1Ff/ssrDYLG7oKvhXwC9+OvH6gz29LA
            zySVUw3GJ55nxDXPvHyuEYFDdvcAtVOtKSpRY4rM28fpKUUP
            Oh5vfIU+ZzIny3xYeuzyGUF4C56hs2Ln0W7qqRA/HD00gquuC
            f16Z5pfBgpt1v1F+4/p5yhkh85KXrLT4Orlc2U=</xenc:CipherVal
            ue>
        </xenc:CipherData>
          <xenc:ReferenceList>
          <xenc:DataReference URI="#Id-
            2h0o8YyBNXw43GDTnEzNJEpV"></xenc:DataReference>
        </xenc:ReferenceList>
      </xenc:EncryptedKey>
    </wsse:Security>
</env:Header>
```

```
<env:Body>
  <xenc:EncryptedData Id="Id-2h0o8YyBNXw43GDTnEzNJEpV"
    Type="http://www.w3.org/2001/04/xmlenc#Element"
    xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
    <xenc:EncryptionMethod
      Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-
      cbc"></xenc:EncryptionMethod>

      <xenc:CipherData>
      <xenc:CipherValue>8N8BVSIRhcczmYh0izngIFk6W+mxqbEDHAg8Y
        5njj2stwKulZoig6UtXYHtRSJCcxYM2X0FxuDYssjqP/LkkrsdxueX
        vkNyeE1huvQ9S9zy1yzOW2Wi7sru7sDzZ3Ctz9HZ7fgRgBS+IYr/
        gCRodvl1kW1zY9bvZ/onYf9KznfNd4fcjYHkrnRlhczFkVaSYzEN0
        mpRRNrGZBP1yGQfVUznxoi7oWIgC</xenc:CipherValue>
    </xenc:CipherData>
  </xenc:EncryptedData>
</env:Body>
</env:Envelope>
```

**Operation Plus**

Submitted at 20 February 2005 17:45:06 o'clock GMT+07:00
Method: Calculator.PlusService.Plus
Arguments:
i : 5
j : 9
CallStack:
Plus()

**Returned from Plus**

Submitted at 20 February 2005 17:45:06 o'clock GMT+07:00
Return value: 14

**Service Response**

Submitted at 20 February 2005 17:45:06 o'clock GMT+07:00

```
<SOAP-ENV:Envelope xmlns:SOAP-
  ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <SOAP-ENV:Body>
    <ns:PlusResponse xmlns:ns="http://www.openuri.org/">
    <ns:PlusResult xmlns:ns="http://www.openuri.org/">14</ns:PlusResult>
  </ns:PlusResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

**5.2 Xml – Signature in Soap**

**Service Request Subtraction**

<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<env:Header>
<wsse:Security env:mustUnderstand="1" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
<wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3" wsu:Id="Id-WcJAkETsqtx4NDtXC_C6q9_6" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">**MIICRTCCAa6gAwIBAAIEPrBeaTANBgkqhkiG9w0BAQUFA**
**DBnMQswCQYDVQQGEwJVUzELMAkGA1UECBMCTlkxFjAUBgNVBA**
**cTDU5ldyBZb3JrIENpdHkxDDAKBgNVBAoTA29yZzETMBEGA1UECx**
**MKY2xpZW50MU9yZzEQMA4GA1UEAxMHY2xpZW50MTAeFw0wM**
**zA0MzAyMzM4MTdaFw0wNDA0MjkyMzM4MTdaMGcxCzAJBgNVBA**
**YTAlVTMQswCQYDVQQIEwJOWTEWMBQGA1UEBxMNTmV3IFlvcm**
**sgQ2l0eTEMMAoGA1UEChMDb3JnMRMwEQYDVQQLEwpjbGllbnQx**
**T3JnMRAwDgYDVQQDEwdjbGllbnQxMIGfMA0GCSqGSIb3DQEBAQ**
**UAA4GNADCBiQKBgQDF/Q/4VGVOb0fdrXELYh1JzKC76eICnJLrCC**
**h6nBfpKjZUBBiDlLhphB52arGonEUIBHHO9n68N1hoN/uz5j6H5/K**
**mLRdcA1huAIlcNoWmxC61XjCxEDT+agvrg2D6suyzElusWCrvpIEs**
**WEtCcCD0x/MOVcQLK3q9oMg4ihj4ewIDAQABMA0GCSqGSIb3DQ**
**EBBQUAA4GBAKgcU99Prrz37UgiTp5NTX4oLDPM+HBmETQB9EnQ**
**PDPZ829tsHsPymM42Pe2Qk4TNM/+ZIdbrFRSft64WWHYjr8K8uB**
**R9F7/a1WyJmiNPE3wkiZlM140HjV8l0fAfwR2d+cdB0RvJpwLx/on**
**TxFcnMlCzJfUUp5mFHzebkw19/WD**
</wsse:BinarySecurityToken>

```
<dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
            <dsig:SignedInfo>
            <dsig:CanonicalizationMethod
              Algorithm="http://www.w3.org/2001/10/xml-exc-
              c14n#"></dsig:CanonicalizationMethod>
            <dsig:SignatureMethod
              Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
              sha1"></dsig:SignatureMethod>
              <dsig:Reference URI="#Id-_kp9Wov0jqVol9r0Lx_kwEuB">
                <dsig:Transforms>
                <dsig:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
                  c14n#"></dsig:Transform>
              </dsig:Transforms>
              <dsig:DigestMethod
                Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"></dsig:Dig
                estMethod>
              <dsig:DigestValue>BoJE8ecX+RLGTNRRnlVVMpdU2kU=</dsig:Di
                gestValue>
            </dsig:Reference>
          </dsig:SignedInfo>
          <dsig:SignatureValue>K+pWzkEZ8U6PhK9yNsB1iEsPEVTDzvMCRasT
            Gvtqx2nXEmcuJkFXkye3gKlVXXBL4OkZrqAKwSCwDVk+eDR8Bg
            qB+vxtwpnt8dbKREEpJfoCny4J5MTfEwWiwZo/wPukprCEjMIlh
            GU/JQ1QKPJprCKqD8T0Ncoj097jIkloF6Q=</dsig:SignatureValue>
            <dsig:KeyInfo>
              <wsse:SecurityTokenReference>
              <wsse:Reference URI="#Id-
                WcJAkETsqtx4NDtXC_C6q9_6"></wsse:Reference>
            </wsse:SecurityTokenReference>
          </dsig:KeyInfo>
        </dsig:Signature>
      </wsse:Security>
    </env:Header>
      <env:Body wsu:Id="Id-_kp9Wov0jqVol9r0Lx_kwEuB"
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
        wssecurity-utility-1.0.xsd">
        <n1:Subtraction xmlns:n1="http://www.openuri.org/">
        <n1:i xmlns:n1="http://www.openuri.org/">10</n1:i>
        <n1:j xmlns:n1="http://www.openuri.org/">7</n1:j>
      </n1:Subtraction>
    </env:Body>
</env:Envelope>
```

**Operation Subtraction**

Submitted at 21 February 2005 00:25:02 o'clock GMT+07:00
Method: Calculator.SubService.Subtraction
Arguments:
i : 10
j : 7
CallStack:
Subtraction()

**Returned from Subtraction**

Submitted at 21 February 2005 00:25:02 o'clock GMT+07:00
Return value: 3

**Service Response**

Submitted at 21 February 2005 00:25:02 o'clock GMT+07:00

```
<SOAP-ENV:Envelope xmlns:SOAP-
  ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <SOAP-ENV:Body>
    <ns:SubtractionResponse xmlns:ns="http://www.openuri.org/">
    <ns:SubtractionResult
      xmlns:ns="http://www.openuri.org/">3</ns:SubtractionResult>
  </ns:SubtractionResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

**5.3 Xml – Signature & Xml - Encryption in Soap**

**Service Request multiply**

<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<env:Header>
<wsse:Security env:mustUnderstand="1" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
<wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3" wsu:Id="Id-Ku3cqdioxWhHqbE214Dm61Ky" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">**MIICRTCCAa6gAwIBAAIEPrBeaTANBgkqhkiG9w0BAQUFA**
**DBnMQswCQYDVQQGEwJVUzELMAkGA1UECBMCTlkxFjAUBgNVBA**
**cTDU5ldyBZb3JrIENpdHkxDDAKBgNVBAoTA29yZzETMBEGA1UECx**
**MKY2xpZW50MU9yZzEQMA4GA1UEAxMHY2xpZW50MTAeFw0wM**
**zA0MzAyMzM4MTdaFw0wNDA0MjkyMzM4MTdaMGcxCzAJBgNVBA**
**YTAlVTMQswCQYDVQQIEwJOWTEWMBQGA1UEBxMNTmV3IFlvcm**
**sgQ2l0eTEMMAoGA1UEChMDb3JnMRMwEQYDVQQLEwpjbGllbnQx**
**T3JnMRAwDgYDVQQDEwdjbGllbnQxMIGfMA0GCSqGSIb3DQEBAQ**
**UAA4GNADCBiQKBgQDF/Q/4VGVOb0fdrXELYh1JzKC76eICnJLrCC**
**h6nBfpKjZUBBiDlLhphB52arGonEUIBHHO9n68N1hoN/uz5j6H5/K**
**mLRdcA1huAIlcNoWmxC61XjCxEDT+agvrg2D6suyzElusWCrvpIEs**
**WEtCcCD0x/MOVcQLK3q9oMg4ihj4ewIDAQABMA0GCSqGSIb3DQ**
**EBBQUAA4GBAKgcU99Prrz37UgiTp5NTX4oLDPM+HBmETQB9EnQ**
**PDPZ829tsHsPymM42Pe2Qk4TNM/+ZIdbrFRSft64WWHYjr8K8uB**
**R9F7/a1WyJmiNPE3wkiZlM140HjV8l0fAfwR2d+cdB0RvJpwLx/on**
**TxFcnMlCzJfUUp5mFHzebkw19/WD**</wsse:BinarySecurityToken>

```
<dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
    <dsig:SignedInfo>
    <dsig:CanonicalizationMethod
      Algorithm="http://www.w3.org/2001/10/xml-exc-
      c14n#"></dsig:CanonicalizationMethod>
    <dsig:SignatureMethod
      Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
      sha1"></dsig:SignatureMethod>
      <dsig:Reference URI="#Id-zQSihyTPST6d5BUy59Ahu9C8">
        <dsig:Transforms>
        <dsig:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
          c14n#"></dsig:Transform>
      </dsig:Transforms>
      <dsig:DigestMethod
        Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"></dsig:Dig
        estMethod>
      <dsig:DigestValue>nnqy3vHeTmRCpXrk+rcL396o7oA=</dsig:Dig
        estValue>
    </dsig:Reference>
  </dsig:SignedInfo>
  <dsig:SignatureValue>kB8C3JPLQXag4hE3iLv2dsVnSwlbdzKaLXpioU
    jJd09FE7kyKHlLaCFicZLWu7DfUvWLgchD76JR9SVFDJqFX3PIHC
    Eui9DWF2qnlyRqQ5Xal2tyDnoxFf6whsDakYjS108D4LLoL0n8la8
    F1X3bWyyhGBoQHvLQkWOD6o5zMXg=</dsig:SignatureValue>
    <dsig:KeyInfo>
      <wsse:SecurityTokenReference>
      <wsse:Reference URI="#Id-
        Ku3cqdioxWhHqbE214Dm61Ky"></wsse:Reference>
    </wsse:SecurityTokenReference>
  </dsig:KeyInfo>
</dsig:Signature>
  <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
  <xenc:EncryptionMethod
    Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-
    1_5"></xenc:EncryptionMethod>
    <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
      <wsse:SecurityTokenReference>
        <dsig:X509IssuerSerial>
        <dsig:X509IssuerName>CN=MyCompany, OU=Development,
          O=MyDevTeam, L=Sealand, ST=WA,
          C=US</dsig:X509IssuerName>
<dsig:X509SerialNumber>1051745810</dsig:X509SerialNumber>
```

```
            </dsig:X509IssuerSerial>
          </wsse:SecurityTokenReference>
        </dsig:KeyInfo>
          <xenc:CipherData>
          <xenc:CipherValue>b3hUXKeqKwLnAQIcHWjkTTk8U5klV5IXhLnm
            ooMX0XjQHJFG31LAyGeYnOKmZBtnEnYTpWG8QyR5chc3cevR
            u+aiqXYf6vy0GLpOySuWqLiWPuTkMnX2DsR8YUeWawln37IX
            R3mOaSh4iyJeTsMcf/whn8rCuFwdPlqv+rySmAQ=</xenc:Ciphe
            rValue>
        </xenc:CipherData>
          <xenc:ReferenceList>
          <xenc:DataReference URI="#Id-
            Q2IdgPcaVOyM2Zy9jobJsc_k"></xenc:DataReference>
        </xenc:ReferenceList>
      </xenc:EncryptedKey>
    </wsse:Security>
  </env:Header>
    <env:Body wsu:Id="Id-zQSihyTPST6d5BUy59Ahu9C8"
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
      wssecurity-utility-1.0.xsd">
      <xenc:EncryptedData Id="Id-Q2IdgPcaVOyM2Zy9jobJsc_k"
        Type="http://www.w3.org/2001/04/xmlenc#Element"
        xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
      <xenc:EncryptionMethod
        Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-
        cbc"></xenc:EncryptionMethod>
        <xenc:CipherData>
        <xenc:CipherValue>LBU+KOOcnUnGhUpwcyAzTIz14K53TWhA0ksMi
          OYAtsLJpfCdiHw5Q2QN0n3iQJwZsAl2yFkqZ392/TmK2Qh7Pav6
          qISdN54fPcwi6/3WAL8TvEHQN6dV+wy1s/MvldX4YXGiWH4Sok
          b0L4nEvcpcxUM1X5neqZNo0vW8ClLi/ujraAqdnzEzbPobQLDAw2
          Hqc3HJyIWNyZNvk9h/h3k0WBdjinoJ307eYwiLOPFqvgU=</xenc
          :CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </env:Body>
</env:Envelope>
```

**Operation multiply**

Submitted at 21 February 2005 10:25:30 o'clock GMT+07:00
Method: Calculator.MultiService.multiply
Arguments:
i : 3
j : 7
CallStack:
multiply()

**Returned from multiply**

Submitted at 21 February 2005 10:25:30 o'clock GMT+07:00
Return value: 21

**Service Response**

Submitted at 21 February 2005 10:25:30 o'clock GMT+07:00

```
<SOAP-ENV:Envelope xmlns:SOAP-
  ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <SOAP-ENV:Body>
    <ns:multiplyResponse xmlns:ns="http://www.openuri.org/">
    <ns:multiplyResult
      xmlns:ns="http://www.openuri.org/">21</ns:multiplyResult>
  </ns:multiplyResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```