

การเข้าถึงข้อมูลเอ็กซ์เอ็มแอลด้วยวิธีแยกองค์ประกอบอิลิเมนต์

Accessing XML Data with Element Separation

พิทักษ์ เอี่ยมประไพ

ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์และเทคโนโลยี

มหาวิทยาลัยธรรมศาสตร์ ศูนย์รังสิต ปทุมธานี 12121

วีระศักดิ์ ชิงถาวร

ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์และเทคโนโลยี

มหาวิทยาลัยธรรมศาสตร์ ศูนย์รังสิต ปทุมธานี 12121

บทคัดย่อ

รูปแบบการเข้าถึงข้อมูลภายในเอกสารเอ็กซ์เอ็มแอลมีความสำคัญเป็นอย่างยิ่ง เนื่องจากภาษาเอ็กซ์เอ็มแอลเป็นภาษาที่ใช้กันอย่างแพร่หลาย ทั้งในด้านการจัดเก็บข้อมูลหรือเป็นรูปแบบในการแลกเปลี่ยนข้อมูล โดยที่การพัฒนาโปรแกรมประยุกต์เพื่อใช้ติดต่อกับเอ็กซ์เอ็มแอลในปัจจุบันยังมีรูปแบบและวิธีการที่ซับซ้อน ซึ่งมีผลต่อความเข้าใจของนักพัฒนา ทำให้การพัฒนาโปรแกรมในส่วนนี้ใช้ระยะเวลาเป็นอันมาก

งานวิจัยนี้จึงได้เสนอ รูปแบบวิธีการในการเข้าถึงข้อมูลภายในเอกสารเอ็กซ์เอ็มแอลด้วยวิธีการแยกองค์ประกอบอิลิเมนต์ เพื่อลดระยะเวลาในการพัฒนาโปรแกรมที่เกี่ยวข้องกับเอกสารเอ็กซ์เอ็มแอลและง่ายต่อการเข้าใจของนักพัฒนา โดยได้พัฒนาเฟรมเวิร์คที่ใช้ในการสร้างชุดคำสั่งในการเข้าถึงข้อมูลภายในเอกสารเอ็กซ์เอ็มแอลด้วยวิธีแยกองค์ประกอบของอิลิเมนต์ พร้อมทั้งประยุกต์ใช้ชุดคำสั่งดังกล่าวกับการเข้าถึงข้อมูลประวัติของโปรแกรมเอ็มเอสเออนแมสเซนเจอร์

คำสำคัญ : ข้อมูลเอ็กซ์เอ็มแอล องค์ประกอบอิลิเมนต์

Abstract

Contemporary, XML has become the most popular language for both exchanging and collecting data throughout the internet. Dueing to its widespread use, the efficiency of data accessing method in XML documents should be primarily concerned. At the present time, the application development based on XML data access is still difficult to implement and need complicated procedures to approach with XML documents, which results in time-consuming development process.

This thesis proposed a solution for XML data access with a methodology called, separation of elemental composition, that resulted in less application development time and was easier for developers to understand. The methodology was implemented by developing a framework for generating AdvAX API which provided the function for accessing data in XML documents by separation of elemental composition method. The API was applied to access MSN Messenger log files.

Keywords : accessing XML data , element separation , AdvAX API

1. บทนำ

รูปแบบการเข้าถึงข้อมูลภายในเอกสารเอ็กซ์เอ็มแอลเป็นวิธีการในการอ่านค่าข้อมูลต่าง ๆ ภายในเอกสารเอ็กซ์เอ็มแอล (XML-Extensible Markup Language) [1] ซึ่งประกอบไปด้วย ชื่อแท็ก (tag name), ชื่อแอตริบิวต์ (attribute name), ค่าแอตริบิวต์ (attribute value), ข้อมูลในอิลิเมนต์ (content), คอมเมนต์ (comment) และโพรเซส ซึ่งอินสตรัคชัน (processing instruction) ในปัจจุบันสามารถแบ่งรูปแบบการเข้าถึงข้อมูลภายในเอกสารเอ็กซ์เอ็มแอลออกเป็น 5 วิธีหลัก ๆ คือ

1.1 Push Model [2]

เป็นรูปแบบ streaming API ที่มี parser เป็นตัวควบคุมโดยที่ parser จะอ่านข้อมูลภายในเอกสารเอ็กซ์เอ็มแอลแล้วส่งเหตุการณ์พร้อมทั้งข้อมูลต่าง ๆ ให้กับโปรแกรมในลักษณะที่เรียกว่า “callback interface” เครื่องมือที่ใช้วิธีการนี้ ได้แก่ แซก (SAX-Simple API for XML) [3], XNI (Xerces Native Interface) [4]

โดยวิธีการนี้จะใช้เวลาในการประมวลผลข้อมูลภายในเอกสารเอ็กซ์เอ็มแอลที่รวดเร็วและใช้หน่วยความจำน้อย อีกทั้งลักษณะการทำงานยังสามารถทำงานได้ก่อนประมวลผลเสร็จทั้งเอกสาร แต่ลักษณะของการพัฒนาโปรแกรมด้วยวิธีนี้มีความยุ่งยากในการเข้าถึงข้อมูลที่มีความซับซ้อนภายในโครงสร้างของเอกสารเอ็กซ์เอ็มแอล เพราะวิธีการนี้ขาดการจัดการเกี่ยวกับระดับชั้นและเงื่อนไขของข้อมูล

1.2 Pull Model [5]

เป็นรูปแบบ streaming API อีกประเภทหนึ่งที่แตกต่างกับ push model ตรงที่โปรแกรมเป็นตัวควบคุมการทำงานโดยถาม parser ถึงเหตุการณ์ที่จะเกิดขึ้น เพื่อเลือกเหตุการณ์ที่สนใจเข้ามาทำงาน เครื่องมือที่ใช้วิธีการนี้ ได้แก่ XMLPULL [6], NekoPull [7], StAX (Streaming API for XML) [8], kXML [9]

โดยวิธีการนี้จะมีประสิทธิภาพใกล้เคียงกับ push model ในด้านเวลาในการประมวลผลและจำนวนหน่วย

ความจำที่ใช้ อีกทั้งการพัฒนาโปรแกรมในรูปแบบนี้สามารถควบคุมเหตุการณ์ที่สนใจได้ แต่ยังคงมีความยุ่งยากในการตรวจสอบระดับชั้นของข้อมูลและรูปแบบในการอ่านข้อมูลที่ต้องการภายในเอกสารเอ็กซ์เอ็มแอลที่มีโครงสร้างของข้อมูลที่ซับซ้อน

1.3 Tree Oriented

เป็นรูปแบบ tree-based API โดย parser จะอ่านข้อมูลภายในเอกสารเอ็กซ์เอ็มแอลทั้งหมดแล้วสร้าง object model ในรูปแบบโครงสร้างต้นไม้ (tree) โดยแต่ละโหนด (node) จะมีการแยกชนิดของข้อมูลที่ชัดเจนตามประเภทของข้อมูล เครื่องมือที่ใช้วิธีการนี้ ได้แก่ DOM (Document Object Model) [10], JDOM [11], DOM4J [12], Sparta [13], XOM (XML Object Model) [14]

โดยวิธีการนี้จะใช้เวลาและหน่วยความจำเป็นจำนวนมาก อีกทั้งรูปแบบในการเข้าถึงข้อมูลเป็นแบบการท่องเข้าไปในโครงสร้างต้นไม้ ซึ่งจะต้องใช้การวนลูปเพื่ออ่านข้อมูลในแต่ละระดับชั้น หากโครงสร้างของข้อมูลมีความซับซ้อนและมีความลึกของต้นไม้ จะต้องทำการวนลูปเป็นจำนวนมาก ทำให้การอ่านข้อมูลที่ต้องการมีความยุ่งยากและซับซ้อน

1.4 Data Binding

เป็นรูปแบบ tree-based API ที่มีชนิดของแต่ละโหนดตามโครงสร้างภายในเอกสารเอ็กซ์เอ็มแอล โดยจะทำการรับเค้าร่างเอ็กซ์เอ็มแอล (XML Schema) [15] เพื่อสร้างชุดคำสั่งในการอ่านข้อมูลภายในเอกสารเอ็กซ์เอ็มแอล ตามเค้าร่างนั้น เครื่องมือที่ใช้วิธีการนี้ ได้แก่ JAXB (Java Architecture for XML Binding) [16], Caster [17]

โดยวิธีการนี้มีรูปแบบการเข้าถึงข้อมูลคล้ายวิธี Tree Oriented แต่ในการเข้าถึงข้อมูลโครงสร้างต้นไม้สามารถเลือกอินเทอร์เฟซที่ตรงกับชื่อของอิลิเมนต์ได้ ทำให้มีรูปแบบที่ชัดเจน และเข้าใจโครงสร้างภายในเอกสารเอ็กซ์เอ็มแอลได้สะดวก แต่วิธีการนี้จะใช้เวลาในการประมวลผลและหน่วยความจำเป็นจำนวนมาก อีกทั้งยังมีการวนลูปเพื่ออ่านข้อมูลที่ต้องการ

1.5 Query API

เป็นรูปแบบ API ที่มีการกำหนด expression หรือ template เพื่อบอกถึงเงื่อนไขและรูปแบบโครงสร้างเอกสาร XML ที่ต้องการเครื่องมือที่ใช้วิธีการนี้ได้แก่ XQuery [18], XPath [19], XSLT (Extensible Stylesheet Language Transformations) [20]

โดยวิธีการนี้จะเป็นการกำหนดเงื่อนไขหรือรูปแบบโครงสร้างเอกสาร XML ที่ต้องการ แล้วทำการประมวลผลข้อมูลภายในเอกสาร XML เพื่อให้ได้ข้อมูลที่ต้องการตามเงื่อนไขหรือรูปแบบโครงสร้างเอกสาร XML ตามที่กำหนด

จากวิธีการที่ได้กล่าวมาทั้งหมดจะเห็นได้ว่ารูปแบบในการเข้าถึงข้อมูลในแต่ละวิธียังมีรูปแบบที่ไม่สะดวกต่อการเข้าถึงข้อมูลที่มีโครงสร้างของข้อมูลที่ซับซ้อน เช่น ภายในเอกสาร XML ที่มีชื่ออิลิเมนต์เดียวกันแต่ความหมายของข้อมูลแตกต่างกันเพราะอยู่ภายในระดับชั้นที่ต่างกันหรือข้อมูลที่ต้องการอยู่ภายใต้โครงสร้างที่ซับซ้อนและมีความลึกมาก

งานวิจัยนี้จึงได้นำเสนอรูปแบบวิธีการในการเข้าถึงข้อมูลภายในเอกสาร XML ที่มีโครงสร้างของข้อมูลที่ซับซ้อนและง่ายต่อการเข้าใจในการพัฒนาโปรแกรม โดยใช้วิธีการเข้าถึงข้อมูลแบบแยกองค์ประกอบของอิลิเมนต์เพื่อสะดวกต่อการเข้าถึงข้อมูลที่ต้องการ ซึ่งวิธีการนี้เป็นรูปแบบ data binding ที่รับคำร้อง XML เพื่อสร้างชุดคำสั่งที่เป็นแบบ streaming API ส่วนที่ 2 อธิบายรูปแบบวิธีการเข้าถึงข้อมูลภายในเอกสาร XML ด้วยวิธีการแยกองค์ประกอบของอิลิเมนต์ ส่วนที่ 3 อธิบายรูปแบบเฟรมเวิร์คในการสร้างชุดคำสั่งสำหรับเข้าถึงข้อมูลภายในเอกสาร XML ด้วยวิธีการแยกองค์ประกอบอิลิเมนต์ รวมทั้งรูปแบบการใช้งานชุดคำสั่งภายในโปรแกรมประยุกต์ ส่วนที่ 4 อธิบายวิธีการประยุกต์ใช้ชุดคำสั่งดังกล่าวกับข้อมูลประวัติการใช้งานของโปรแกรมเอ็มเอสเอ็นเอ็มเอสเซนเจอร์ และวัดประสิทธิภาพในด้านเวลาในการประมวลผลและหน่วยความจำที่ใช้ในการประมวลผลเทียบกับ SAX ส่วนที่ 5 สรุปผลการทดลองวิธีการเข้าถึง

ข้อมูลภายในเอกสาร XML ด้วยวิธีการแยกองค์ประกอบและข้อเสนอแนะ

2. รูปแบบวิธีการเข้าถึงข้อมูลภายในเอกสาร XML ด้วยวิธีการแยกองค์ประกอบของอิลิเมนต์

การแยกองค์ประกอบของอิลิเมนต์เป็นรูปแบบวิธีการเข้าถึงข้อมูล โดยมีชุดคำสั่ง (Java API) ที่ใช้ในการอ่านข้อมูล ซึ่งภายในชุดคำสั่งประกอบไปด้วยคลาสของอิลิเมนต์ที่รับลงทะเบียนและอินเทอร์เฟซของอิลิเมนต์ที่ใช้ลงทะเบียน สามารถอธิบายได้ดังนี้

2.1 คลาสรับลงทะเบียน

เป็นคลาสที่รับลงทะเบียนอินเทอร์เฟซของอิลิเมนต์ โดยมีการกำหนดชื่อของคลาสและตำแหน่งของคลาสในแพ็คเกจ (package) เหมือนกับอินเทอร์เฟซแต่ชื่อคลาสจะตามด้วยคำว่า “EventHandler” เพื่อบอกว่าเป็นคลาสที่รับอินเทอร์เฟซของอิลิเมนต์นั้น ๆ

2.2 อินเทอร์เฟซที่ใช้ลงทะเบียน

เป็นอินเทอร์เฟซที่ใช้ในการลงทะเบียนเพื่อเลือกความต้องการอ่านข้อมูลจากอิลิเมนต์ใด โดยมีการกำหนดชื่อของอินเทอร์เฟซ, ฟังก์ชันภายในอินเทอร์เฟซ และตำแหน่งของอินเทอร์เฟซภายในแพ็คเกจ สามารถอธิบายได้ดังนี้

1. ชื่อของอินเทอร์เฟซ จะตั้งชื่อเดียวกับชื่อของ อิลิเมนต์ ตามด้วยคำว่า “Listener” เพื่อบอกว่าเป็นอินเทอร์เฟซในการอ่านข้อมูลภายในอิลิเมนต์นั้น

2. ฟังก์ชันภายในอินเทอร์เฟซ จะตั้งชื่อฟังก์ชันตามชื่อแอตทริบิวต์ของอิลิเมนต์นั้น ๆ ถ้ามีค่าข้อมูลภายในอิลิเมนต์จะตั้งชื่อฟังก์ชันเป็นชื่อเดียวกับชื่ออิลิเมนต์ เพื่ออ่านข้อมูลภายในอิลิเมนต์นั้น โดยจะมีค่านำหน้าฟังก์ชันเป็นชื่ออิลิเมนต์ที่ดูแลอิลิเมนต์ (root node) ถึงอิลิเมนต์ตัวเอง คั่นแต่ละอิลิเมนต์ด้วยเครื่องหมาย “_” เพื่อให้ทราบว่าฟังก์ชันนี้สำหรับรับค่าของแอตทริบิวต์หรือค่าภายในอิลิเมนต์ที่ระดับของชั้นข้อมูลใด

3. ตำแหน่งของอินเทอร์เฟซภายในแพ็คเกจ การกำหนดตำแหน่งของอินเทอร์เฟซภายในแพ็คเกจจะ

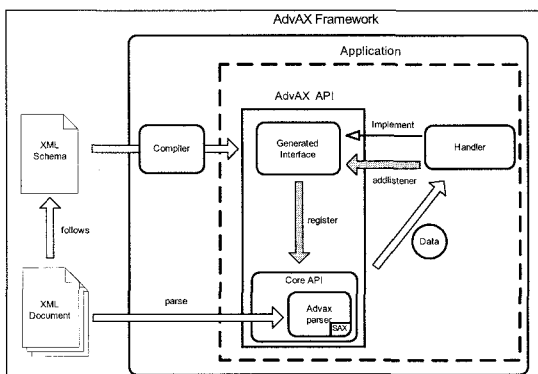
กำหนดตามระดับชั้นของข้อมูล โดยแพ็คเกจจะขึ้นต้นด้วยคำว่า “advax” ตามด้วยชื่อของรูทอิลิเมนต์จนถึงอิลิเมนต์พื่อ (parent node) ซึ่งวิธีนี้จะทำให้การเข้าถึงข้อมูลภายในอิลิเมนต์ที่ต้องการมีความชัดเจนและเป็นรูปแบบเดียวกัน เพราะโครงสร้างของแพ็คเกจจะตรงตามโครงสร้างภายในเอกสารเอ็กซ์เอ็มแอล

ดังนั้นงานวิจัยนี้ได้เสนอวิธีการเข้าถึงข้อมูลภายในเอกสารเอ็กซ์เอ็มแอลโดยการสร้างชุดคำสั่งที่เป็นแบบ streaming API ซึ่งในการพัฒนาโปรแกรมสามารถเลือกอินเทอร์เฟซที่ใช้ในการเข้าถึงข้อมูลของอิลิเมนต์ที่ต้องการได้โดยตรง ทำให้การพัฒนาสะดวก และมีรูปแบบที่ชัดเจนและง่ายต่อการเข้าใจในการอ่านค่าข้อมูลในส่วนที่ต้องการ

3. รูปแบบการทำงานของระบบ

อธิบายถึงเฟรมเวิร์คในการสร้างชุดคำสั่งสำหรับการเข้าถึงข้อมูลภายในเอกสารเอ็กซ์เอ็มแอลและการทำงานของชุดคำสั่งด้วยวิธีแยกองค์ประกอบอิลิเมนต์ภายในโปรแกรมประยุกต์ สามารถอธิบายได้ดังนี้

3.1 การทำงานของ AdvAX Framework

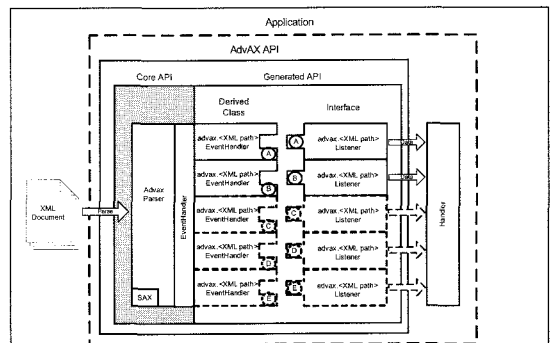


รูปที่ 1 รูปแบบการทำงานของ AdvAX Framework

ในงานวิจัยนี้ได้เสนอเฟรมเวิร์คที่ชื่อว่า AdvAX (Advance API for XML) ซึ่งเป็นเฟรมเวิร์คที่รับเคำร่างเอ็กซ์เอ็มแอลผ่านคอมไพเลอร์ (compiler) แล้วสร้างชุดคำสั่งที่ใช้ในการเข้าถึงข้อมูลภายในเอกสารเอ็กซ์เอ็มแอล

ด้วยวิธีแยกองค์ประกอบอิลิเมนต์ โดยเรียกชุดคำสั่งนี้ว่า “AdvAX API” ภายในชุดคำสั่งนี้จะประกอบไปด้วย 2 ส่วนคือ อินเทอร์เฟซและคลาสที่ตรงตามวิธีแยกองค์ประกอบของอิลิเมนต์ และส่วนของชุดคำสั่งหลักที่มี Advax parser ในการอ่านข้อมูลภายในเอกสารเอ็กซ์เอ็มแอล โดยที่การทำงานภายใน Advax parser จะใช้ SAX ในการอ่านข้อมูลต่าง ๆ ภายในเอกสารเอ็กซ์เอ็มแอล

3.1 การทำงานของ AdvAX API



รูปที่ 2 รูปแบบการทำงานของ AdvAX API

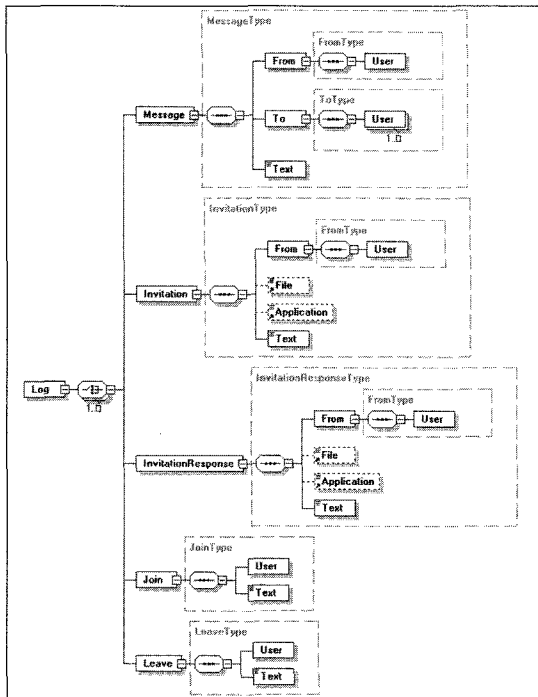
การทำงานภายในโปรแกรม นักพัฒนาจะสร้าง handler ที่ได้ implement อินเทอร์เฟซของอิลิเมนต์ที่ต้องการ แล้วทำการลงทะเบียนกับคลาสที่รับลงทะเบียนของอินเทอร์เฟซนั้น ๆ แล้วนำคลาสนั้นไปขึ้นทะเบียนกับคลาส EventHandler ที่อยู่ในชุดคำสั่งหลัก จากนั้นจะทำการอ่านข้อมูลภายในเอกสารเอ็กซ์เอ็มแอลด้วย Advax parser จากนั้น Advax parser จะส่งข้อมูลไปให้กับ EventHandler ทำการแยกข้อมูลที่ต้องการส่งให้กับ Handler ที่ได้ implement อินเทอร์เฟซนั้น ๆ ไว้

4. การใช้งานชุดคำสั่ง และวัดประสิทธิภาพในการประมวลผล

4.1 การประยุกต์ใช้งานชุดคำสั่ง AdvAX API

งานวิจัยนี้วิธีการเข้าถึงข้อมูลภายในเอกสารเอ็กซ์เอ็มแอลจะต้องมีเคำร่างเอ็กซ์เอ็มแอลของเอกสารเอ็กซ์-

เอ็มแอลนั้น โดยในกรณีศึกษาการเข้าถึงข้อมูลภายใน log ไฟล์ของโปรแกรม MSN Messenger มีโครงสร้างภายในเอกสารเอกซ์เอ็มแอล ดังรูปที่ 3



รูปที่ 3 โครงสร้างเอกสารเอกซ์เอ็มแอล log file ของโปรแกรม MSN Messenger

จากโครงสร้างของเอกสารเอกซ์เอ็มแอล จะแสดงวิธีการเข้าถึงข้อมูลของอิลิเมนต์ User ที่อยู่ในอิลิเมนต์ Invitation โดยมีคลาสและอินเทอร์เฟซที่ได้จาก AdvAX Framework โดยชุดคำสั่งต่างๆ จะอยู่ใน AdvAX API ที่ได้จากการคอมไพล์เค้าร่างเอกซ์เอ็มแอล สามารถแสดงได้ดังนี้

- ส่วนการสร้าง handler ที่ใช้ในการเข้าถึงข้อมูล

```
public class handler implements advax.log.invitation.from.UserListener{
    public void log_invitation_from_user_FriendlyName(String data) {
        System.out.println("User name : "+data);
    }
}
```

- ส่วนของการลงทะเบียน handler กับ

EventHandler

```
org.advax.core.EventHandler eh = new org.advax.core.EventHandler();
advax.log.invitation.from.UserEventHandler ueh = new
advax.log.invitation.from.UserEventHandler();
Handler h = new Handler();
ueh.addListener(h);
eh.addListener(ueh);
```

- ส่วนของการ parse เอกสารเอกซ์เอ็มแอล

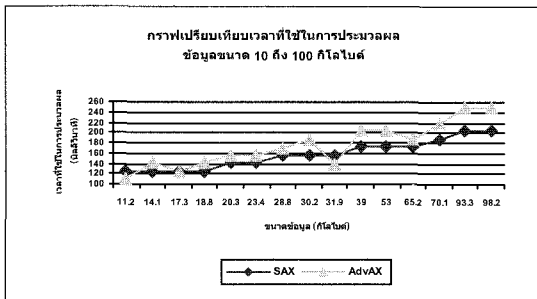
```
org.advax.core.AdvaxParser.parse("log.xml", eh);
```

จะเห็นได้ว่าการเข้าถึงข้อมูลรูปแบบนี้ สามารถแยกส่วนการทำงานกับกลุ่มของข้อมูลได้ โดยสร้าง handler ขึ้นมาหลาย ๆ ตัวเพื่อจัดการกับกลุ่มของข้อมูลในส่วนต่าง ๆ ที่ต้องการ ทำให้รูปแบบในการพัฒนาสามารถกระจายงานโดยแยกส่วนการทำงานได้ ซึ่งการลงทะเบียนจะติดตั้งเพียงครั้งเดียว และการเข้าถึงข้อมูลด้วยวิธีแยกองค์ประกอบอิลิเมนต์สามารถเข้าถึงข้อมูลได้โดยตรงผ่านทางฟังก์ชันภายในอินเทอร์เฟซของอิลิเมนต์ที่ต้องการ ทำให้มีรูปแบบที่แน่นอน ชัดเจน ซึ่งเป็นผลให้นักพัฒนาสามารถเข้าใจได้ตรงกัน อีกทั้งรูปแบบของอินเทอร์เฟซยังมีความสอดคล้องกับโครงสร้างภายในเอกสารเอกซ์เอ็มแอล ทำให้การเลือกใช้งานอินเทอร์เฟซของอิลิเมนต์ที่ต้องการทำได้สะดวก

4.2 การวัดประสิทธิภาพเปรียบเทียบกับ SAX

การวัดประสิทธิภาพในการประมวลผลข้อมูลภายในเอกสารเอกซ์เอ็มแอลจะเริ่มต้นตั้งแต่เริ่มต้นอ่านเอกสารเอกซ์เอ็มแอลจนถึงสิ้นสุดการอ่านเอกสารเอกซ์เอ็มแอล โดยข้อมูลที่ใช้ทดสอบมีขนาดตั้งแต่ 10 ถึง 100 กิโลไบต์ บนเครื่องคอมพิวเตอร์ Intel Pentium 4 ความเร็ว 1.7 GHz. หน่วยความจำ 1 GHz. ระบบปฏิบัติการ WindowXP บนระบบจาวาเวอร์ชวลแมชีนเวอร์ชัน 1.4.7

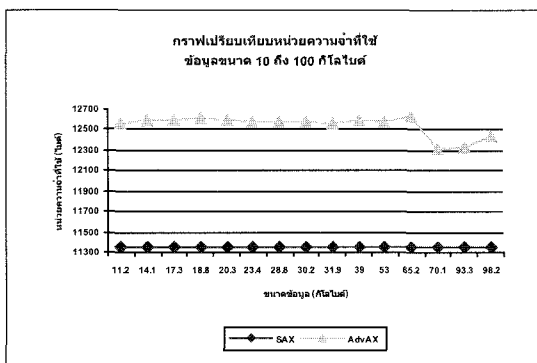
• เวลาในการประมวลผลข้อมูล



รูปที่ 4 กราฟแสดงเวลาที่ใช้ในการประมวลผลข้อมูล

จากรูปที่ 4 จะเห็นได้ว่า SAX ใช้เวลาในการประมวลผลเฉลี่ยประมาณ 157.27 มิลลิวินาที และ AdvAX ใช้เวลาในการประมวลผลเฉลี่ยประมาณ 175.93 มิลลิวินาที คิดเป็น 111.86 % ของเวลาที่ SAX ใช้ในการประมวลผล

• หน่วยความจำที่ใช้ในการประมวลผลข้อมูล



รูปที่ 5 กราฟแสดงหน่วยความจำในการประมวลผลข้อมูล

จากรูปที่ 5 จะเห็นได้ว่า SAX ใช้หน่วยความจำในการประมวลผลเฉลี่ยประมาณ 11,360 ไบต์ และ AdvAX ใช้หน่วยความจำในการประมวลผลเฉลี่ยประมาณ 12,544.53 ไบต์ คิดเป็น 110.43 % ของเวลาที่ SAX ใช้ในการประมวลผล

จากผลการทดลองสามารถสรุปได้ว่า AdvAX เป็นเครื่องมือที่ใช้ในการเข้าถึงข้อมูลภายในเอกสารอิเล็กทรอนิกส์ที่พัฒนาจาก SAX และทำงานบนระบบจาวา ซึ่งใช้เวลาและหน่วยความจำในการประมวลผลข้อมูลมากกว่า SAX ประมาณ 10% เพราะ AdvAX จะมีส่วนที่จัดการการเข้าถึงข้อมูลในแต่ละระดับชั้นและแยกข้อมูลต่าง ๆ ในแต่ละอิลิเมนต์ โดยมีการสร้างชุดคำสั่งที่ใช้ในการพัฒนาโปรแกรมประยุกต์เพื่อเข้าถึงข้อมูลที่ต้องการได้อย่างมีประสิทธิภาพและสะดวกกว่า SAX รวมทั้งมีรูปแบบการเข้าถึงข้อมูลที่มีความชัดเจนและง่ายต่อความเข้าใจของนักพัฒนา ทำให้งานวิจัยนี้เป็นอีกวิธีการหนึ่งในการเข้าถึงข้อมูลภายในเอกสารอิเล็กทรอนิกส์ที่มีประสิทธิภาพและสะดวกต่อการใช้งาน

5. สรุปผลการวิจัย

ในงานวิจัยนี้ได้เสนอรูปแบบการเข้าถึงข้อมูลภายในเอกสารอิเล็กทรอนิกส์ด้วยวิธีแยกองค์ประกอบของอิลิเมนต์ที่เป็นรูปแบบ data-binding โดยวิธีการเข้าถึงข้อมูลอยู่ในรูปแบบ push model ซึ่งเป็น streaming api ทำให้การประมวลผลข้อมูลใช้เวลาและหน่วยความจำน้อย โดยการพัฒนาโปรแกรมส่วนที่ติดต่อกับอิเล็กทรอนิกส์ด้วยวิธีการนี้สามารถอ่านข้อมูลภายในเอกสารอิเล็กทรอนิกส์โดยมีรูปแบบที่แน่นอน ชัดเจนและมีความเข้าใจตรงกัน ซึ่งมีการแบ่งแยกระดับชั้นของข้อมูลและส่วนประกอบภายในอิลิเมนต์ รูปแบบการเข้าถึงข้อมูลด้วยวิธีนี้จะเหมาะสมสำหรับโครงสร้างของเอกสารอิเล็กทรอนิกส์ที่มีโครงสร้างแบบซับซ้อน เช่น ข้อมูลที่ต้องการอยู่ในระดับที่ลึก หรือข้อมูลอยู่ภายในอิลิเมนต์ที่มีชื่อซ้ำกัน แต่ความหมายต่างกันได้สะดวกมากยิ่งขึ้น

การเข้าถึงข้อมูลด้วยวิธีการแยกองค์ประกอบของอิลิเมนต์นี้ สามารถพัฒนาเพิ่มในส่วนของข้อมูลที่เป็นแบบ mix-content ที่ข้อมูลภายในอิลิเมนต์มีทั้งแท็กเป็นส่วนประกอบอยู่ด้วย รวมทั้งสามารถพัฒนาการทำงานของคอมไพเลอร์ในการแปลงรูปแบบของเค้าร่างอิเล็กทรอนิกส์-

แอลที่มีความหลากหลายเป็นชุดคำสั่งที่ใช้ในการเข้าถึงข้อมูล

6. เอกสารอ้างอิง

- [1] World Wide Web Consortium. XML, <http://www.w3.org/TR/2004/REC-xml-20040204/>
- [2] Aleksander Slominiski, "Design of a Pull and Push Parser System for Streaming XML", Indiana University, http://www.extreme.indiana.edu/xgws/papers/xml_push_pull/
- [3] D. Megginson et al. Sax 2.0, The Simple API for Xml, <http://www.saxproject.org/>
- [4] The Apache XML Project. XNI, Xerces Native Interface, <http://xml.apache.org/xerces2-j/xni.html/>
- [5] Aleksander Slominiski, "On Using XML Pull Parsing Java APIs", Indiana University, 2004, <http://www.xmlpull.org/history/>
- [6] Common API for XML Pull Parsing <http://www.xmlpull.org/>
- [7] Andy Clark, "CyberNeko Pull Parser", NekoPull <http://www.apache.org/~andyc/neko/doc/pull/>
- [8] Java Community Process, StAX, JSR 173: Streaming API for XML, <http://jcp.org/en/jsr/detail?id=173/>
- [9] Stefan Haustein, kXML Project, <http://www.kxml.org/>
- [10] World Wide Web Consortium. DOM, <http://www.w3.org/dom/>
- [11] Jason Hunter, Brett McLaughlin, JDOM, <http://www.jdom.org/>
- [12] James Strachan, Maarten Coene, DOM4J, <http://www.dom4j.org/>
- [13] Eamonn O' Brien-Strain , Sparta , <http://sparta-xml.sourceforge.net/>
- [14] Elliott Rusty Harold, XOM, XML Object Model API, <http://www.cafeconleche.org/XOM/>
- [15] World Wide Web Consortium, XML Schema, <http://www.w3.org/XML/Schema/>
- [16] Sun Microsystems, JAXB, Java Architecture for XML Binding, <http://java.sun.com/xml/jaxb/>
- [17] Arnaud Blandin, Caster, <http://www.castor.org/>
- [18] World Wide Web Consortium, XQuery, <http://www.w3.org/XML/Query>
- [19] World Wide Web Consortium, XPATH, <http://www.w3.org/TR/xpath>
- [20] World Wide Web Consortium, XSLT, Extensible Stylesheet Language Transformations, <http://www.w3.org/TR/xslt>