ภาคผนวก ข

เทคนิคการเขียนโปรแกรมที่ใช้ในการพัฒนา SAFWA

ในส่วนของภาคผนวก ข กล่าวถึงเทคนิคการเขียนโปรแกรมที่ใช้สำหรับการพัฒนา SAFBWA ที่น่าสนใจ เพื่อเป็นแนวทางสำหรับการพัฒนาโปรแกรมประยุกต์หรืองานอื่นต่อไป โดยจะพูดถึงเทคนิค การเขียนโปรแกรมการส่งข้อมูลบนเครือข่าย การเขียนโปรแกรมแบบมัลติเธรดดิ้ง (Multithreading) การเขียนโปรแกรมเพื่อจัดการแฟ้มข้อมูล XML โดยรวมถึงการแปลงเอกสาร XML เป็นเอกสารรูปแบบ HTML

ข.1 การเขียนโปรแกรมเพื่อส่งข้อมูลบนเครือข่าย

สำหรับการติดต่อสื่อสารบน SAFWA ส่งข้อมูลบนเครือข่ายโดยใช้โพรโตคอล TCP/IP โดยอาศัย API ของจาวาคือ java.net เป็นการสื่อสารแบบ socket based มองข้อมูลที่เข้ามาเป็นสายของข้อมูล (data stream) มองการรับส่งข้อมูลจากเครือข่ายเหมือน การอ่านเขียนข้อมูลจากแฟ้มข้อมูล โดยวาจาจัดแบบการส่งข้อมูลไว้สองแบบคือ stream socket และ datagram socket โดยแบบแรกจะเป็นการเชื่อมต่ออยู่ตลอด (connection oriented) และแบบที่สองเป็นแบบไม่ต้องมีการเชื่อมต่อ (connectionless oriented) โดยการพัฒนา SAFWA เลือกใช้แบบ stream socket การทำงานของเอเจนท์ในการติดต่อสื่อสารมีสองแบบคือ การรับฟัง และการส่งข้อความ

การเขียนโปรแกรมเพื่อรอรับข้อความสามารถทำได้โดยใช้ 5 ขั้นตอนดังนี้

1. สร้าง ServerSocket

```
ServerSocket server = new ServerSocket(port,queuelength);
```

โดยที่ port คือ port ที่เปิดสำหรับรับการติดต่อ และ
queuelength เป็นความยาวของข้อความที่รอรับได้

2. รอรับข้อมูลที่จะถูกส่งเข้ามา

```
Socket connection = server.accept();
```

3. สร้างตัวแปรสำหรับรองรับสายข้อมูลที่ส่งมาผ่านเข้ามา

```
ObjectInputStream input = new ObjectInputStream(connection.getInputStream);
```

4. นำเอา input ที่ได้ไปประมวลผลตามที่ต้องการ
5. ทำการปิดการรับข้อความ

```
connection.close();
```

การเขียนโปรแกรมเพื่อส่งข้อความมีขั้นตอนดังนี้คือ

1. สร้างการเชื่อมต่อไปยังเครื่องที่จะส่งข้อความ

```
Socket  connection = new Socket(serveraddr,port);
```

โดยที่ serveraddr คือ IP address ของเครื่องปลายทาง และ
port คือ port ที่เปิดไว้สำหรับรอรับข้อความ

2. ส่งข้อความโดยผ่านตัวแปร output

```
ObjectOutputStream output = new
ObjectUutputStream(connection.getOutputStream);
```

3. ปิดการเชื่อมต่อ

```
connection.close();
```

สำหรับการเขียนโปรแกรมทั่วไปสามารถนำมาใช้กับการรับและส่งข้อมูลพร้อมกันได้ ซึ่งอยู่กับการออกแบบการทำงานว่าต้องการทำงานแบบใด

ข.2 การเขียนโปรแกรมแบบมัลติเธรดดิ้ง (Multithreading)

ประโยชน์ของการเขียนโปรแกรมแบบมัลติเธรดดิ้ง คือแต่ละเธรดสามารถทำงานไปพร้อมกันได้โดยไม่ส่งผลกระทบต่อกัน อาศัยคลาส Thread ที่อยู่ในจาวา API java.lang

สำหรับการเขียนโปรแกรมมัลติเธรดสามารถสร้างคลาสที่เราต้องการให้ทำงานแบบมัลติเธรดดิ้งได้โดย เพิ่มคำสั่งดังนี้

```
public class Test extends Thread {
        ....
        public void run() {
                ..../// ชุดคำสั่ง
        }
}
```

จากการเขียนคลาสด้านบนชุดคำสั่งที่ต้องการให้ทำงานจะถูกเขียนลงในเมธอด run() และเมื่อต้องการให้เธรดทำงานสามารถเรียกได้โดยคำสั่ง

```
Test.start();
```

เมธอด run() ที่ถูกเขียนขึ้นเป็นการ overriding เมธอด run() ที่มีอยู่ในคลาส Thread เมื่อเรียกเมธอด start() เธรดก็จะทำงานโดยทำตามชุดคำสั่งที่อยู่ในเมธอด run() โดยอัตโนมัติ

ข.3 การเขียนโปรแกรมแบบเพื่อจัดการเอกสาร XML

การพัฒนา SAFWA ผู้พัฒนาอาศัย API สำหรับจัดการเอกสาร XML ที่ชื่อ dom4j โดยสามารถดาว์นโหลด และศึกษาวิธีการใช้งานได้จาก www.dom4j.org

ในส่วนภาคผนวกนี้จะแนะนำเกี่ยวกับการสร้างเอกสาร XML การจัดการข้อมูลในเอกสาร การอ่านและการเขียนเอกสาร XML จากแฟ้มข้อมูล และการจัดรูปแบบเอกสาร XML

ข.3.1 การสร้างเอกสาร XML

การสร้างเอกสาร XML โดยอาศัย dom4j สามารถทำได้ดังนี้คือ

```
Document doc = DocumentHelper.createDocument();
```

เมื่อสร้างเอกสาร XML แล้ว จะต้องทำการสร้าง root element

```
Element root = doc.addElement(rootName);
```

ตัวอย่างเอกสาร XML ที่ต้องการสร้าง ตัวอย่างเช่นต้องการสร้างรายการสินค้าโดยการเก็บ รหัสสินค้า ชื่อสินค้า วันที่สั่ง จำนวนที่สั่ง และราคา

```
<products>
        <item id="">
                <name/>
                <date/>
                <amount/>
                <price/>
        </item>
</product>
```

ทำการสร้าง root element ดังนี้คือ

```
Element root = doc.addElement("products");
```

จากคำสั่งจะได้เอกสาร XML รูปแบบดังนี้

```
<products/>
```

ข.3.2 การจัดการเอกสาร XML

การจัดการเอกสาร XML คือการ เพิ่มข้อมูล และการค้นหาข้อมูลในเอกสาร XML

ข.3.2.1 การเพิ่มข้อมูลลงในเอกสาร XML

การเพิ่มข้อมูลลงในเอกสาร XML จะต้องเพิ่ม สามารถเพิ่มได้ในรูปของ Element Attribute และ Text โดยที่ถ้าต้องการเพิ่ม Element สามารถทำได้ดังนี้คือ

```
Element item = root.addElement("item");
```

ได้เอกสาร XML ดังนี้

```
<products>
        <items/>
</product>
```

ทำการเพิ่ม Attribute id ใน Element item ทำได้ดังนี้คือ

```
item.addAtribute("id","001");
```

ได้เอกสาร XML ดังนี้

```
<products>
        <items id = 001/>
</product>
```

จากนี้สามารถเพิ่ม Element อื่นๆ ได้ดังนี้คือ

```
item.addElement("name").addText("IBM R40");
```

ได้เอกสาร XML ดังนี้

```
<products>
        <items id = 001>
                <name>IBM R40</name>
        </item>
</product>
```

ข.3.2.2 การดึงข้อมูลจากเอกสาร XML

การดึงข้อมูลจากเอกสาร XML นำมาใช้งานสามารถทำได้โดยอาศัย Xpath ในการระบุ Element ที่ต้องการสามารถเลือกดึงได้ทั้งหมดหรือเลือกดึงเพียง Element เดียวก็ได้

ถ้าเป็นการดึงข้อมูลเพียงตัวเดียว อย่างเช่นต้องการชื่อสินค้า ที่มีรหัส 001 สามารถทำได้ดังนี้คือ

```
XPath xpathSelector = DocumentHelper.createXPath(path);
String result = xpathSelector.selectSingleNode(doc).getText();
```

ตัวอย่างสำหรับ path คือ

```
/product/item[@id='001']
```

สำหรับการดึงข้อมูลเป็นชุดจะต้องอาศัยอินเตอร์เฟส List ใน java.util เพื่อนำข้อมูลออกมาในรูปแบบของ List ดังนี้คือ

```
String path = "/product/item";
XPath xpathSelector = DocumentHelper.createXPath(path);
List result = xpathSelector.selectNodes(doc);
```

เมื่อได้ข้อมูลในรูปแบบ List สามารถนำข้อมูลมาแสดงได้ดังนี้คือ

```
Iterator iter = result.iterator();
while (iter.hasNext()){
        Element item = (Element) iter.next();
        String id = item.attribute("id").getText();
        String name = item.element("name").getText();
        String date = item.element("date").getText();
        String amount = item.element("amount").getText();
        String price = item.element("price").getText();
        …
}
```

ข.3.2.3 การอ่านข้อมูลเอกสาร XML จากแฟ้มข้อมูล

ในการตั้งค่าเริ่มต้นของเอเจนท์ได้เก็บอยู่ในรูปแบบแฟ้มข้อมูลดังนั้นการนำมาใช้งานก็จะต้องอ่านเอกสาร XML จากแฟ้มข้อมูลทำได้ดังนี้คือ อาศัย API จาก org.dom4j.io

```
SAXReader xmlReader = new SAXReader();
Document doc = xmlReader.read(aFile);
```

ซึ่งจะได้เอกสาร doc ในรูปแบบ XML ไปใช้งาน

ข.3.2.4 การเขียนข้อมูลเอกสาร XML ไปยังแฟ้มข้อมูล

การข้อมูลในรูปแบบ XML ลงในแฟ้มข้อมูลหรือส่งผ่านไปยังเครือข่ายในรูปแบบของสายข้อมูลได้ดังนี้คือ

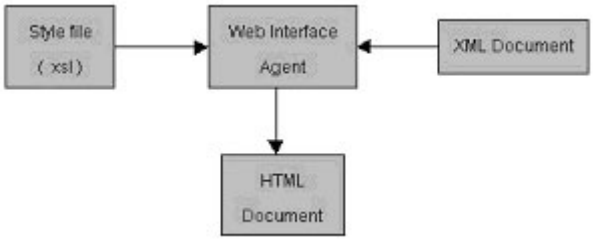
```
OutputFormat outformat = OutputFormat.createPrettyPrint();
outformat.setEncoding("UTF-8");
XMLWriter writer = new XMLWriter(out,outformat);
```

```
writer.write(doc);
writer.flush();
```

โดยที่จัดรูปแบบเอกสารโดยใช้ตัวเข้ารหัสอักษรแบบ UTF-8 โดยที่ out แทน output stream ซึ่งเป็นได้ทั้งลงแฟ้มข้อมูล และส่งผ่านเครือข่าย

ข.3.2.5 การจัดรูปแบบเอกสาร XML

การจัดรูปแบบเอกสาร XML สามารถทำได้โดยการแปลงเอกสารที่อยู่ในรูปแบบเอกสาร XML ไปยังรูปแบบอื่น เช่นเอกสาร HTML รูปแบบการแปลงเอกสารแสดงในภาพประกอบ ข.1



ภาพประกอบ ข.1 แสดงแผนภาพการแปลงเอกสาร XML เป็น HTML

ในแผนภาพ ส่วนของ Web interface agent มีชุดคำสั่งในการแปลงเอกสารรูปแบบ XML เป็น HTML โดยการอ่านรูปแบบจากแฟ้ม style.xsl ตัวอย่างสำหรับเอกสารรูปแบบ XML มีดังนี้

```
<product>
        <item id=001>
                <name>IBM R40</name>
                <date>10/08/2003</date>
                <amount>5</amount>
                <price>50,000</price>
        </item>
        <item id=002>
                <name>Sony Vaio V505</name>
                <date>05/07/2003</date>
                <amount>3</amount>
                <price>80,000</price>
```

```
        </item>
    </product>
```

โดยที่ชุดคำสั่งสำหรับแปลงเอกสารมีดังนี้

```
    TransformerFactory factory = TransformerFactory.newInstance();
    Transformer transformer =
        factory.newTransformer(new StreamSource( stylesheet ) );
    DocumentSource source = new DocumentSource( document );
    DocumentResult result = new DocumentResult();
    transformer.transform( source, result );
    Document transformedDoc = result.getDocument();
```

ตัวแปร stylesheet คือ แฟ้มข้อมูลที่เก็บรูปแบบการจัดเอกสารไว้ document คือ เอกสารที่ต้องการจัดรูปแบบ ส่วนเอกสารที่จัดรูปแบบแล้วจะเก็บที่ transformDoc

สำหรับแฟ้ม style.xsl มีรูปแบบการจัดเอกสารโดยใช้เทคโนโลยี XLST ดังตัวอย่างต่อไปนี้

```
    <xsl:for-each select="/products">
        <xsl:for-each select="item">
        <font size=3 color=blue>
            <b><xsl:value-of select="name"/></b></font>
        <font size=2>
            price = <xsl:value-of select="name"/> baht</font>
            <br/>
        </xsl:for-each>
    </xsl:for-each>
```

จากตัวอย่าง ข้างต้นเมื่อผ่านการจัดรูปแบบ จะได้เอกสารที่ถูกจัดรูปแบบแล้วดังนี้คือ

```
    IBM R40 price = 50,000 baht
    Sony Vaio V505 price = 50,000 baht
```