การพัฒนาโปรแกรมจัดการ FIX Message ด้วยคิวแบบ First In First Out ร่วมกับฐานข้อมูล MySQL
กรณีศึกษา ตลาดหลักทรัพย์แห่งประเทศไทย พ.ศ. 2556

Development of Message Handling Program Using Queue (First In First Out) and MySQL Database
: A CASE STUDY THE STOCK EXCHANGE OF THAILAND

นายวิชุนันท์ เฉลิมเมือง

โครงงานสหกิจศึกษานี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิทยาศาสตร์บัณฑิต สาขาเทคโนโลยีสารสนเทศ
คณะเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีไทย-ญี่ปุ่น
พ.ศ. 2556

คณะกรรมการสอบ

..... ประธานกรรมการสอบ
(อาจารย์.....)

..... กรรมการสอบ
(อาจารย์.....)

..... อาจารย์ที่ปรึกษา
(อาจารย์.....)

..... ประธานสหกิจศึกษาสาขาวิชา
(อาจารย์.....)

ชื่อโครงงาน การ พัฒนาโปรแกรมจัดการ FIX Message ด้วยคิวแบบ First In First Out ร่วมกับฐานข้อมูล MySQL
กรณีศึกษาตลาดหลักทัพย์แห่งประเทศไทย

ผู้เขียน นายวิชุนันท์ เฉลิมเมือง

คณะวิชา คณะเทคโนโลยีสารสนเทศ สาขาระบบสารสนเทศทางธุรกิจ

อาจารย์ที่ปรึกษา นางธันยพร กณิกนันต์

พนักงานที่ปรึกษา 1. นายอภิโชค ประสิทธิ์ศิริผล

2. นางสาวนารีมล ทองศรีคำ

ชื่อบริษัท ตลาดหลักทรัพย์แห่งประเทศไทย

ประเภทธุรกิจ ศูนย์กลางการซื้อขายหลักทรัพย์จดทะเบียน และพัฒนาระบบต่าง ๆ ที่จำเป็นเพื่ออำนวยความสะดวกในการซื้อขายหลักทรัพย์

บทสรุป

จากการเข้ามาปฏิบัติงานในตลาดหลักทรัพย์แห่งประเทศไทย ได้มีโอกาสในการศึกษาระบบงานหลังการซื้อขายหลักทรัพย์ ความต้องการในปรับปรุงระบบงาน และได้มีส่วนร่วมในการพัฒนาระบบเพื่อลดค่าใช้จ่ายที่ค่อนข้างสูงของระบบเดิม โดย ส่วนที่นักศึกษาได้รับมอบหมายให้รับผิดชอบคือ เขียนโปรแกรมทดสอบการรับ-ส่งข้อมูล ธุรกรรมของการซื้อ-ขายหลักทรัพย์ โดยใช้ Library ที่เรียกว่า QuickFIX และฐานข้อมูล MySQL ในการจัดการการทำงานระหว่าง Server ที่เป็น .NET Platform และ Client ที่เป็น Java Platform เพื่อทำการทดสอบและศึกษาว่ามีประสิทธิภาพพอที่จะนำมาใช้แทนระบบเดิมได้หรือไม่

ผลที่ได้จากการพัฒนาโปรแกรม โปรแกรมสามารถทำงานร่วมกันได้และสามารถจัดการรับส่งข้อมูลได้ตามวัตถุประสงค์ที่ได้กำหนดไว้ และในฐานะที่เป็นส่วนหนึ่งในการพัฒนาระบบ ประโยชน์ที่ได้รับคือ ได้เรียนรู้ภาษาโปรแกรมที่ไม่เคยใช้งานมาก่อน รวมถึงได้เรียนรู้ประสบการณ์การทำงานจริงในองค์กร

กิตติกรรมประกาศ

โครงการสหกิจศึกษาฉบับนี้สามารถสำเร็จลุล่วงไปได้ดี เนื่องจากได้รับความกรุณาและความช่วยเหลืออย่างดีจากพี่ ๆ หลาย ๆ ท่าน รวมถึงตลาดหลักทัพย์แห่งประเทศไทยที่ให้โอกาสได้มาร่วมฝึกปฏิบัติงานในครั้งนี้

ขอขอบคุณ นายอภิโชค ประเสริฐศิริผล และ นางสาวนารีมล ทองศรีคำ ที่เป็นพนักงานที่ปรึกษา และพี่ ๆ ทุกท่านในฝ่ายงานบริการหลังการซื้อขายหลักทรัพย์ ที่ได้สอนการทำงาน และมอบประสบการณ์การทำงานมากมายให้ตลอดระยะเวลา 4 เดือนที่ปฏิบัติงาน นอกจากนี้ขอขอบพระคุณผู้เกี่ยวข้องทุกท่านที่มีส่วนช่วยให้รายงานฉบับนี้เสร็จสมบูรณ์โดยเฉพาะ นาง ธันยพร กณิกนันต์ ที่เป็นอาจารย์ที่ปรึกษาในการจัดทำรายงานฉบับนี้

สารบัญ

หน้าที่

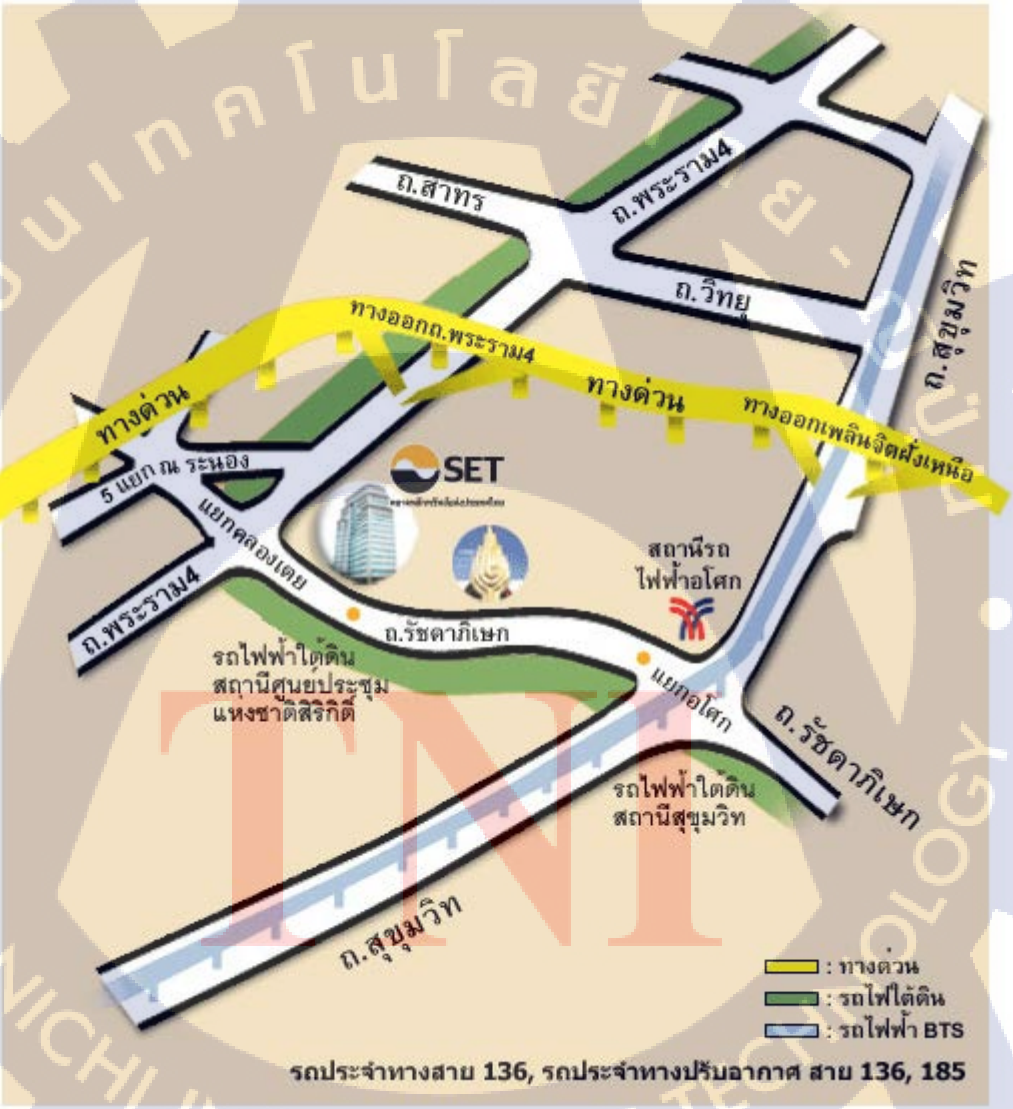
สารบัญตาราง

สารบัญรูปประกอบ

บทที่ 1
บทนำ

1.1 ชื่อที่ตั้งและสถานประกอบการ

ตลาดหลักทรัพย์แห่งประเทศไทย ตั้งอยู่ที่ 62 ถนนรัชดาภิเษก เขตคลองเตย กรุงเทพมหานคร 10110



รูปที่ 1.1 แผนที่ตลาดหลักทรัพย์

1.2 ลักษณะธุรกิจของสถานประกอบการ หรือการให้บริการหลักขององค์กร

การดำเนินงานหลัก ได้แก่ การรับหลักทรัพย์จดทะเบียนและดูแลการเปิดเผยข้อมูลของบริษัทจดทะเบียน การซื้อขายหลักทรัพย์และการกำกับดูแลการซื้อขายหลักทรัพย์ การกำกับดูแลบริษัทสมาชิกส่วนที่เกี่ยวข้องกับการซื้อขายหลักทรัพย์ ตลอดจนถึงการเผยแพร่ข้อมูลและการส่งเสริมความรู้ให้แก่ผู้ลงทุน

1.3 รูปแบบการจัดองค์กรและการบริหารองค์กร

พระราชบัญญัติหลักทรัพย์และตลาดหลักทรัพย์ พ.ศ. 2535 กำหนดให้การดำเนินงานของตลาดหลักทรัพย์แห่งประเทศไทย อยู่ภายใต้การกำกับดูแลของคณะกรรมการกำกับหลักทรัพย์และตลาดหลักทรัพย์ (ก.ล.ต.) และกำหนดอำนาจหน้าที่ให้คณะกรรมการตลาดหลักทรัพย์แห่งประเทศไทย เป็นผู้กำหนดนโยบายและควบคุมการดำเนินงานของตลาดหลักทรัพย์ฯ

ตลาดแรก

คณะกรรมการกำกับหลักทรัพย์และตลาดหลักทรัพย์ (ก.ล.ต.) ทำหน้าที่กำกับและดูแลตลาดแรก โดยบริษัทใดที่ต้องการออกหลักทรัพย์ใหม่เสนอขายหุ้นต่อประชาชนครั้งแรก (Initial Public Offering) หรือเสนอขายหลักทรัพย์อื่นๆ แก่ประชาชน ต้องขออนุมัติจากคณะกรรมการกำกับหลักทรัพย์และตลาดหลักทรัพย์ (ก.ล.ต.) และดำเนินการตามเกณฑ์ที่กำหนด จากนั้นคณะกรรมการกำกับหลักทรัพย์และตลาดหลักทรัพย์จะต้องตรวจสอบสถานะทางการเงินและการดำเนินงานของบริษัทนั้นก่อนที่จะอนุมัติให้บริษัททำการออกหลักทรัพย์ขายแก่ประชาชนได้

ตลาดรอง

หลังจากการเสนอขายหุ้นต่อประชาชนครั้งแรก หลักทรัพย์จะสามารถทำการซื้อขายในตลาดรองได้ก็ต่อเมื่อผู้ออกหลักทรัพย์นั้นได้ยื่นคำขอและได้รับอนุมัติจากตลาดหลักทรัพย์แห่งประเทศไทยแล้ว

รูปที่ 1.2 รูปแบบการจัดองค์กรในตลาดหลักทรัพย์

1.4 ตำแหน่งและหน้าที่งานที่นักศึกษาได้รับมอบหมาย

ตำแหน่ง นักศึกษาฝึกงาน แผนก Post-Trade System หน้าที่และลักษณะงานที่ได้รับมอบหมาย คือ เขียนโปรแกรมตามความต้องการที่กำหนด และจัดทำคู่มือประกอบการใช้งาน

1.5 พนักงานที่ปรึกษาและตำแหน่งของพนักงานที่ปรึกษา

1. นายอภิโชค ประสิทธิ์ศิริผล ตำแหน่ง Project Application Engineer
2. นางสาวนารีมล ทองศรีคำ ตำแหน่ง Deputy Head Post-Trade System

1.6 ระยะเวลาที่ปฏิบัติงาน

เริ่มปฏิบัติงานตั้งแต่วันที่ 3 มิถุนายน พ.ศ. 2556 ถึง วันที่ 4 ตุลาคม 2556 รวมทั้งสิ้นเป็นระยะเวลา 4 เดือน

1.7 ที่มาและความสำคัญของปัญหา

ปัจจุบัน ระบบ Transmit Message ในตลาดหลักทรัพย์ได้ใช้ MQ และ BizTalk เป็นตัวกลาง (Middleware) ในการจัดการการรับ-ส่ง Message ต่าง ๆ ซึ่งมีค่าใช้จ่ายในการดำเนินงานค่อนข้างสูง เพื่อลดค่าใช้จ่ายในการดำเนินงานส่วนนี้ลง จึงได้นำซอฟแวร์และเทคโนโลยีอื่น ๆ ที่มีความสามารถเทียบเท่าหรือมีประสิทธิภาพมากกว่ามาใช้ทดแทนระบบเดิม ซึ่งซอฟต์แวร์ที่เลือกมา

ใช้ทำการทดสอบนี้ก็คือ FIX Engine (QuickFIX) กับ MySQL โดยทั้งสองซอฟต์แวร์นี้เป็น Open Source ที่สามารถ Download มาพัฒนาต่อยอดได้

1.8 วัตถุประสงค์หรือจุดมุ่งหมายของโครงงาน

การเขียนโปรแกรมจัดการ FIX Message ด้วยคิวแบบ FIFO (First-In First-Out) และฐานข้อมูล MySQL ได้แยกเป็นหัวข้อย่อย ๆ ดังตารางที่ 1.1

ที่	วัตถุประสงค์
1	เขียนโปรแกรม FIX Application ให้เชื่อมต่อกันระหว่าง Server ที่เป็น .NET และ Client ที่เป็น Java
2	เขียนโปรแกรมทดสอบรับค่า SessionID ที่ Logon มายัง Server แล้วนำมา Insert ลงฐานข้อมูล และ Delete เมื่อ Client Logout เพื่อเป็นการบอกว่ามี Client ใดบ้างที่เชื่อมต่อกับ Server อยู่ในขณะนั้น
3	เขียนโปรแกรมทดสอบการส่ง FIX Message จาก Client ไปให้ Server
4	เขียนโปรแกรมทดสอบการส่ง FIX Message ไปให้ Client โดยให้ Server ดึง Record ที่ได้กำหนดเงื่อนไขไว้จากฐานข้อมูลมาสร้างเป็น FIX Message แล้วส่งไป
5	เขียนโปรแกรมทดสอบส่ง FIX Message จำนวนมาก ๆ จาก Server ไปที่ Client ด้วยหลักการของคิวแบบ FIFO (First-In First-Out) โดยใช้ฐานข้อมูล MySQL เป็นตัวกลางในการจัดการ
6	เขียนโปรแกรมแก้ไขปัญหาเกี่ยวกับการจัดการการส่ง FIX Message นั้น ๆ หาก Client ออฟไลน์ขณะที่ส่ง FIX Message ไปให้
7	เขียนโปรแกรมแก้ไขปัญหาเกี่ยวกับการจัดการ FIX Message หากส่ง FIX Message ไปแล้วเกิด Session Not Found
8	เขียนโปรแกมทดสอบการรับ FIX Message จาก Client มา Insert ลงฐานข้อมูล MySQL

ตารางที่ 1.1 วัตถุประสงค์ของโครงงาน

1.9 ผลที่คาดว่าจะได้รับจากการปฏิบัติงานหรือโครงงานที่ได้รับมอบหมาย

เข้าใจโครงสร้างและหลักการทำงานของ FIX Protocol และสามารถเขียนโปรแกรม FIX Application ร่วมกับโปรแกรมฐานข้อมูล MySQL เพื่อจัดการการรับ-ส่ง FIX Message จำนวนมาก

ๆ ของ Server ที่เป็น .NET Platform และ Client ที่เป็น Java Platform โดยใช้หลักการของคิว (Queue) แบบ First In First Out ได้ตามความต้องการดังตารางที่ 1.1

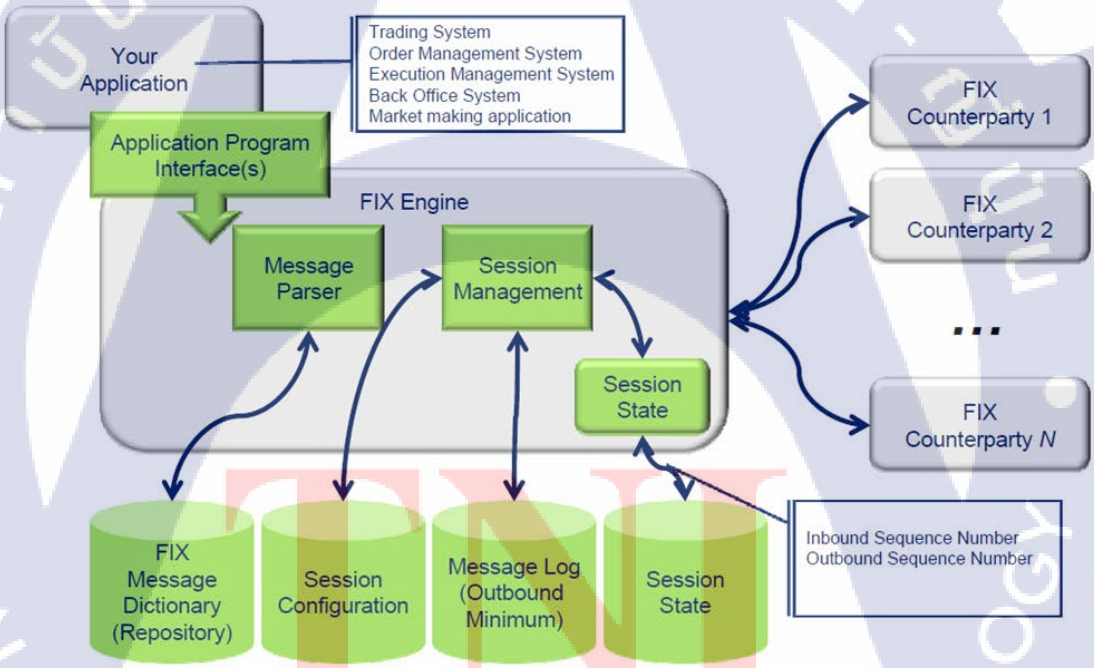
1.10 นิยามศัพท์เฉพาะ

FIX Protocol, QuickFIX, FIX Message

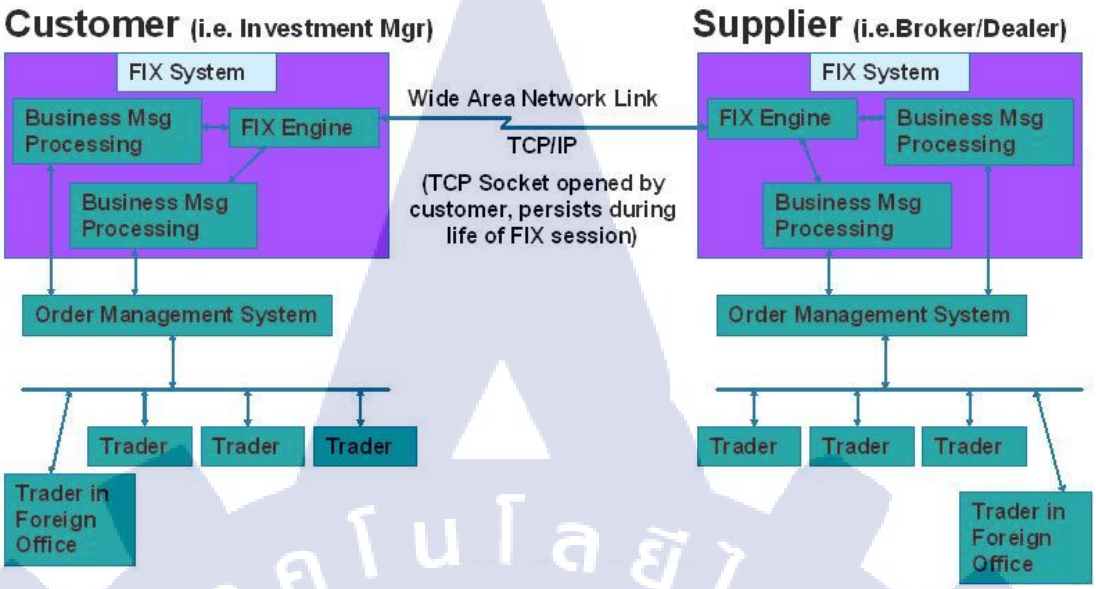
บทที่ 2
ทฤษฎีและเทคโนโลยีที่ใช้ในการปฏิบัติงาน

2.1 FIX Protocol

FIX Protocol ย่อมาจาก Financial Information eXchange Protocol เป็นมาตรฐานการสื่อสารของข้อมูลการซื้อขายหลักทรัพย์แบบเรียลไทม์ (Real Time) ปัจจุบันเป็นโปรโตคอลที่ใช้กันอย่างแพร่หลายในอุตสาหกรรมตลาดการเงิน เช่น สถาบันการเงินและการลงทุน บริษัทหลักทรัพย์โบรกเกอร์/ดีลเลอร์ (Broker/Dealers) และ ระบบอิเล็คโทรนิคสำหรับจัดระบบและตัดการทำธุรกรรมในการซื้อ-ขายทรัพย์สินทางการเงิน เป็นต้น และผลิตภัณฑ์ทางการเงินที่ FIX Protocol รองรับ เช่น ตราสารหนี้ ตราสารทุน ตราสารอนุพันธ์ อัตราแลกเปลี่ยนเงินตราระหว่างประเทศ



รูปที่ 2.1 สถาปัตยกรรมของ FIX



รูปที่ 2.2 ระบบการเชื่อมต่อของ FIX

2.2 รูปแบบของ FIX Message (FIX Message Format)

FIX Message มีองค์ประกอบ 3 ส่วนดังนี้

Header เป็นส่วนที่ใช้ระบุ Message Type, Message Length, Sender/Destination, Sending Time, Sequence Number, Encoding ของ FIX Message

Body เป็นส่วนที่ใช้บรรจุค่า Session และเนื้อหาใน FIX Message ที่ส่งมาจาก Application ต่าง ๆ

Trailer/Footer เป็นส่วนที่ใช้บรรจุค่า Checksum ซึ่งอยู่ลำดับสุดท้ายใน FIX Message เสมอ

```
8=FIX.4.1^9=0235^35=D^34=10^43=N^49=VENDOR^50=CUSTOMER^
56=BROKER^52=19980930-09:25:58^1=XQCCFUND^11=10^21=1^
55=EK^48=277461109^22=1^54=1^38=10000^40=2^44=76.750000^
59=0^10=165
```

Header: 8 (version), 9 (body length), 35 (MsgType), 34 (MsgSeqNum), 43 (PossDupFlag), 49 (SenderCompID), 115 (OnBehalfOfCompID), 56 (TargetCompID), 52 (time stamp)

Body: 1 (Account), 11 (ClOrdID), 21 (HandInst), 55 (Symbol), 48 (SecurityID), 22 (IDSource), 54 (Side), 38 (OrderQty), 49 (OrdType), 44 (Price), 59 (TimeInForce)

Trailer: 10 (Checksum)

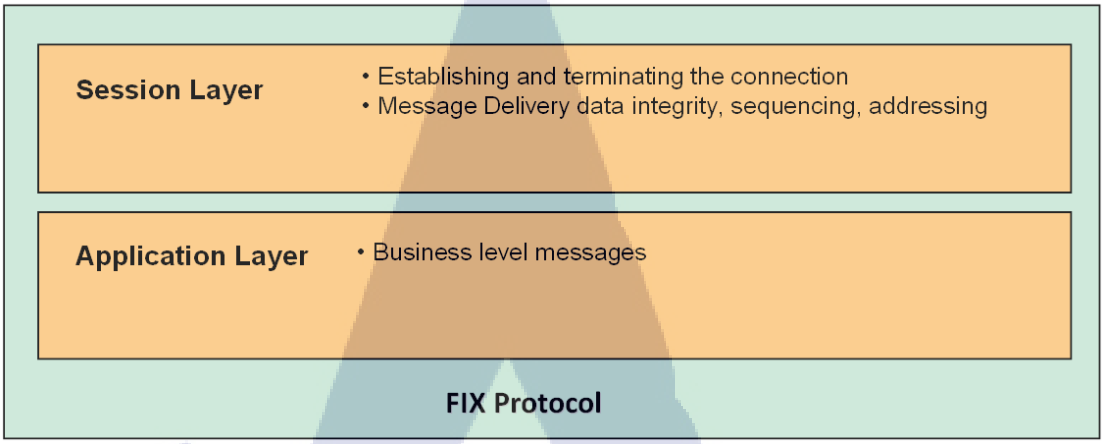
รูปที่ 2.3 Header Body และ Trailer/Footer ใน FIX Message

จากรูปที่ 2.3 จะเห็นว่า FIX Message นั้นเป็นชุดข้อมูลที่ประกอบด้วยฟิลด์ (Field) หลาย ๆ ฟิลด์เรียงต่อกัน แต่ละฟิลด์มีรูปแบบ **<TAG>=<VALUE><DELIMITER>** (หมายเลขแท็ก (Tag) เครื่องหมายเท่ากับ ค่าของแท็ก และตัวคั่น)

ทุก ๆ FIX Message เริ่มต้นด้วยฟิลด์ 8=FIX.x.y^ ซึ่งบอกถึงเวอร์ชั่นของ FIX Protocol ที่ใช้ในการสื่อสารกันระหว่าง Server กับ Client และปิดท้ายด้วยฟิลด์ 10 =nnn^ เสมอซึ่งคือค่า Checksum

2.3 Session Layer และ Application Layer ใน FIX (FIX Session Layer and Application Layer)

ใน FIX ตั้งแต่แวอร์ชันที่ 5.0 เป็นต้นไป Session Layer และ Application Layer จะแยกส่วนและหน้าที่ออกจากกัน ดังรูปที่ 2.4



รูปที่ 2.4 หน้าที่ของ Session Layer และ Application Layer ใน FIX Protocol

2.4 ประเภทของ FIX Message (FIX Message Types)

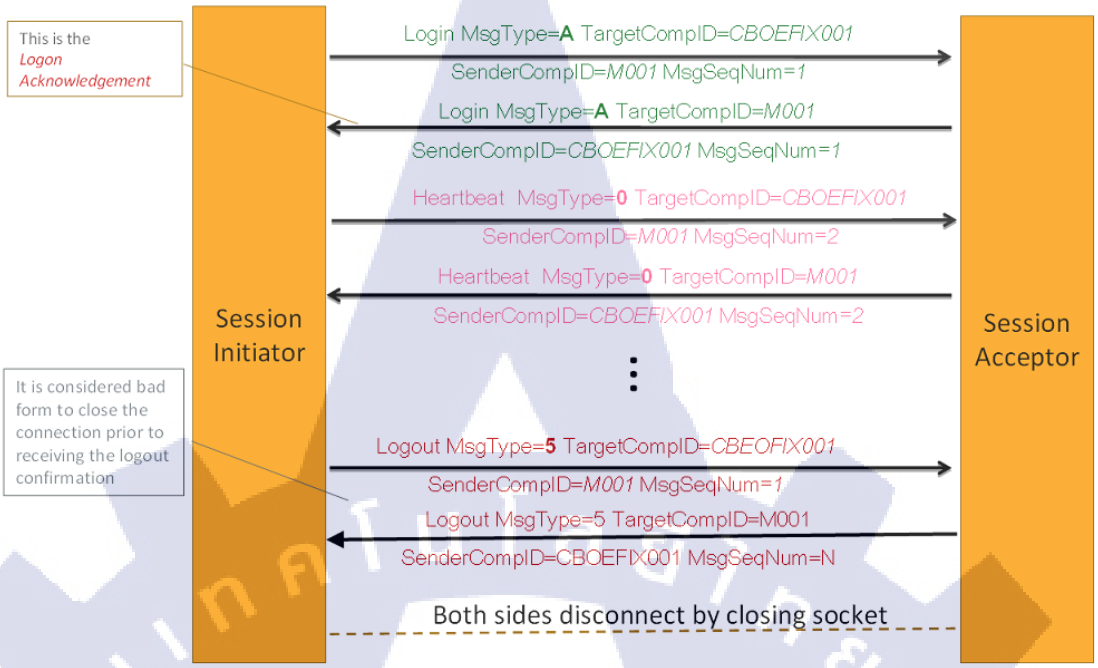
แบ่งออกเป็นสองแบบ คือ Session Message และ Application Message ดังรูปที่ 2.6

Session Messages	*Application Messages*	
Heartbeat	Advertisement	New Order - List
Logon	Indications of Interest	List Status
Test Request	News	List Execute
Resend Request	New Order - Single	List Cancel Request
Reject	Execution Report	List Status Request
Sequence Reset	Order Cancel/Replace	Quote
Logout	Order Cancel Request	Quote Request
	Order Cancel Reject	Quote Cancel
XML_non_FIX	Order Status Request	Email
	Allocation	Settlement Instr.
	Allocation ACK	…
	Trade Capture Report	…

รูปที่ 2.5 ประเภทของ FIX Message

Session Message ใช้สำหรับจัดการเซสชันของระบบที่กำลังเชื่อมต่อกันระหว่าง Server กับ Client เช่น Logon, Logout, Heartbeat

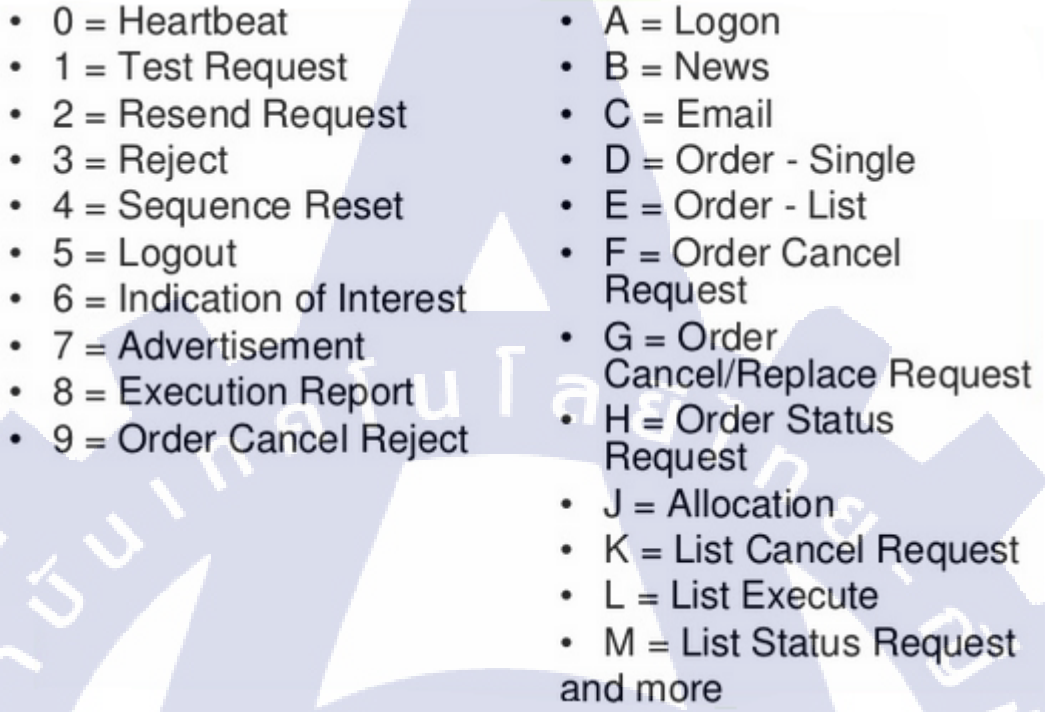
Application Message ใช้บอกถึงประเภทและรายละเอียดต่าง ๆ ของ FIX Message ว่าต้องการทำธุกรรมอะไร เช่น New Single Order, Execution Report



รูปที่ 2.6 แผนผังแสดงลำดับขั้นตอนการรับ-ส่ง FIX Message ใน Session Layer

รูปที่ 2.7 แผนผังแสดงลำดับขั้นตอนการรับ-ส่ง FIX Message ใน Application Layer

2.5 Message Type (แท็ก 35)



รูปที่ 2.8 ตัวอย่าง Message Type (แท็ก 35)

2.6 FIXML (Financial Information Exchange Markup Language)

Syntax ที่เป็น Standard ของ FIX Message มีรูปแบบคือ <Tag>=<Value><Delimiter> ตามที่ได้อธิบายไว้ในหัวข้อที่ 2.5 ดังนั้น ในการส่ง FIX Message เราจะต้องรู้ความหมายของหมายเลขแท็กต่าง ๆ ซึ่งมีมากกว่า 1300 แท็ก จึงไม่สะดวกหาก FIX Message นั้นมีการระบุจำนวนแท็กหรือรายละเอียดลงไปมาก ๆ ดังนั้น วิธีที่ช่วยแก้ปัญหานี้ก็คือการส่งข้อมูลเป็นแบบ XML แนบไปกับ FIX Message ด้วย

XML เป็นภาษาที่เราสามารถกำหนดแท็กหรือความหมายเอง และยังมี XML Parser ให้ใช้งานมากมายในการอ่านค่า และเป็นภาษามาตรฐานที่ใช้แลกเปลี่ยนข้อมูลกันระหว่าง Platform อยู่แล้ว การส่ง FIX Message ที่แนบข้อมูล XML ไปด้วย จะต้องกำหนด MsgType (แท็ก 35) เป็น n และมี Required Fields คือ XmlDataLen (แท็ก 212) และ XmlData (แท็ก 213) โดย XmlData จะถูกกำหนดข้อมูลเข้าไปในแท็ก 213 และ XmlDataLen คือ จำนวนความยาวของข้อมูล ซึ่งทั้งสองฟิลด์นี้จะต้องสัมพันธ์กัน แต่การส่งแต่ละแบบนั้นก็มีข้อดีข้อด้อยต่างกันไปซึ่งอธิบายไว้ในหัวข้อที่ 2.7 และ 2.8

8=FIXT.1.1^9=144^35=n^34=11^49=CLIENT2^52=20130205-
06:45:37.000^56=EXECUTOR^212=55^213=<FIXML><symbol>BBL</symbol><price>44.60</price></FIXML>
^553=test2^554=444^10=245^

Header

8=FIXT.1.1^
9=144^
35=n^
34=11^
49=CLIENT2^
52=20130205-06:45:37.000^
212=55^
213=<FIXML><symbol>BBL</symbol><price>44.60</price></FIXML>
56=EXECUTOR^

Body

Trailer

10=245 Checksum

รูปที่ 2.9 ตัวอย่าง Message Type n (XML non FIX)

8=FIX.4.2^9=251^35=D^49=AFUNDMGR^56=ABROKER^34=2^ 52=20030615-
01:14:49^11=12345^21=1^ 55=IBM^54=1^
60=2003061501:14:49^38=5000^40=2^44=110.75^10=127

Header fields:
8=BeginString (indicates FIX 4.2)
9=BodyLength
35=MsgType (new order)
49=SenderCompID (AFUNDMGR)
56=TargetCompID (ABROKER)
34=MsgSeqNum (2)
52=SendTime

Trailer Fields:
10=Checksum

Body fields:
11=ClOrderID (client order id)
21=HandleInst (automated exec)
55=Symbol (IBM)
54=Side (buy)
56=TransactTime
38=OrderQty (5000)
40=OrdType (Limit)
44=Price (110.75)
52=SendTime

รูปที่ 2.10 Buy 5000 IBM @ 110.75 โดยระบุ MsgType เป็น New Order (35=D)

```
<FIXML>
<FIXMLMessage>
 <Header>
  <SendingTime>20030615-
   01:14:49</SendingTime>
  <Sender>
    <CompID>AFUNDMGR
    </CompID>
  </Sender>
  <Target>
    <CompID>ABROKER
    </CompID>
  </Target>
 </Header>
<ApplicationMessage>
 <Order>
  <ClOrdID>12345
  </ClOrdID>
  <HandlInst Value="1"/>
  <Instrument>
    <Symbol>IBM
    </Symbol>
  </Instrument>
  <Side Value="1"/>
  <TransactTime>
   2003061501:14:49
  </TransactTime>
    <OrderQtyData>
      <OrderQty>5000
      </OrderQty>
    </OrderQtyData>
    <OrdType Value="2"/>
    <Price>110..75</Price>
  </Order>
 </ApplicationMessage>
</FIXMLMessage>
</FIXML>
```

รูปที่ 2.11 Buy 5000 IBM @ 110.75 โดยระบุ MsgType เป็น XML non FIX (35=n)

2.7 ข้อดีของ FIXML

FIXML สามารถใช้ XML Parser ต่าง ๆ ในอ่านค่าได้ เนื่องจากมีข้อกำหนดและคุณสมบัติต่าง ๆ เหมือนภาษา XML แต่ถ้าเป็น FIX Message แบบ Traditional จะต้องมีใช้ FIX engine ในการตรวจสอบและอ่านข้อมูล

Syntax ของ FIXML มีลักษณะ Human Readable คือสามารถอ่านและตีความความหมายของ Message ได้ง่ายกว่า เมื่อเปรียบเทียบกับ FIX Message ที่เป็นแบบ Traditional อ่านและตีความความหมายของ Message ได้ง่ายกว่า () compared with traditional FIX.

FIXML นั้นมี Middleware ที่รองรับมากมาย เนื่องจากเป็นภาษากลางในการแลกเปลี่ยนข้อมูล

2.8 ข้อด้อยของ FIXML

FIXML ต้องใช้ Bandwidth ในการประมวลผลมากกว่า FIX Message ที่เป็นแบบ Traditional

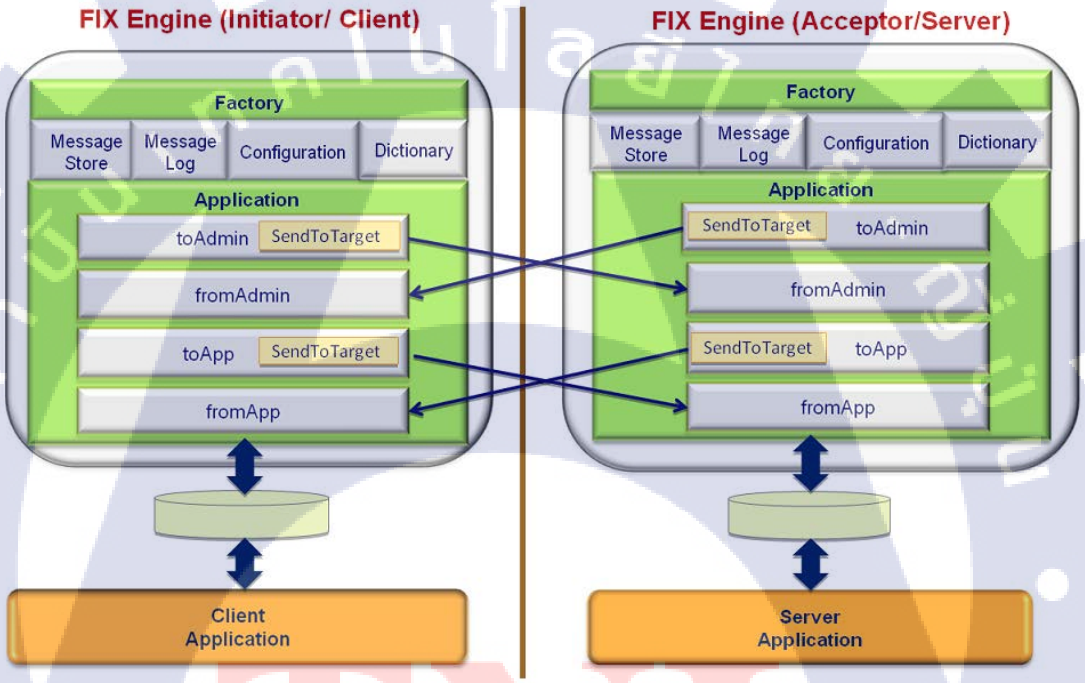
2.9 QuickFIX

QuickFIX เป็น Open Source ในการจัดการ FIX Message หรือเขียน FIX Application โดยสามารถบันทึกข้อมูลการทำงานลงใน Log File อ่านค่า FIX Message จัดการการเชื่อมต่อกัน

ระหว่างเครื่อง และเป็น Cross-Platform สามารถทำงานได้ทั้งระบบปฏิบัติการ Windows, Mac OS X, Linux, Solaris และ Free BSD

สำหรับ QuickFIX (.DLL) ที่ใช้ Implement บน .NET (QuickFIX/n) สามารถดาวน์โหลดได้ที่เว็บไซต์ http://www.quickfixn.org และสำหรับ QuickFIX (.JAR) ที่ใช้ Implement บน Java (QucikFIX/J) สามารถดาวน์โหลดได้ที่เว็บไซต์ http://www.quickfixj.org

2.10 โครงสร้างของ QuickFIX



รูปที่ 2.12 โครงสร้างของ QuickFIX

2.11 การเขียนโปรแกรม FIX Engine (QuickFIX) ด้วย C# และ Java

C#

```
public interface IApplication
{
    void FromAdmin(Message message, SessionID sessionID);
    void FromApp(Message message, SessionID sessionID);
    void OnCreate(SessionID sessionID);
    void OnLogon(SessionID sessionID);
    void OnLogout(SessionID sessionID);
    void ToAdmin(Message message, SessionID sessionID);
    void ToApp(Message message, SessionID sessionId);
}
```

Java

```java
public interface Application {
    void onCreate(SessionID sessionId);
    void onLogon(SessionID sessionId);
    void onLogout(SessionID sessionId);
    void toAdmin(Message message, SessionID sessionId);
    void toApp(Message message, SessionID sessionId)
        throws DoNotSend;
    void fromAdmin(Message message, SessionID sessionId)
        throws FieldNotFound, IncorrectDataFormat,
        IncorrectTagValue, RejectLogon;
    void fromApp(Message message, SessionID sessionId)
        throws FieldNotFound, IncorrectDataFormat,
        IncorrectTagValue, UnsupportedMessageType;
}
```

การเขียน Application โดยใช้ QuickFIX ถ้าเขียนด้วย .NET จะต้องทำการ Implements Interface IApplication หรือถ้าเขียนด้วย Java ก็จะต้อง Implements Interface Application โดยแต่ละ Method จะถูกเรียกใช้เมื่อมีการทำงานดังนี้

FromApp ทุก ๆ Application Message ที่ถูกรับมาจากอีกฝั่งสำเร็จ Method นี้จะถูกเรียกใช้งานขึ้น ตัวอย่าง Message ที่เป็น Application Message เช่น Orders, Executions, Security Definitions และ Market data เป็นต้น

FromAdmin ทุก ๆ Session Message ที่ถูกรับมาจากอีกฝั่งสำเร็จ Method นี้จะถูกเรียกใช้งานขึ้น ตัวอย่าง Message ที่เป็น Session Message เช่น Heartbeat, Logon, และ Logout

OnCreate Method นี้จะถูกเรียกใช้งานเมื่อมีการสร้าง Session สำเร็จ

OnLogon Method นี้จะถูกเรียกใช้งานเมื่อมีการเชื่อมต่อกันระหว่างเครื่องสำเร็จ

OnLogout Method นี้จะถูกเรียกใช้งานหลังจากที่มีการร้องขอทำการ Logout และทำการ Logout สำเร็จแล้ว หรือเครื่องใดที่มีหลุดออกจากการเชื่อมต่อไป Method นี้ก็จะทำงานเช่นกัน

ToApp หลังจากที่ Application Message ถูกส่งออกไป Method นี้จะถูกรียกใช้งาน

ToAdmin เมื่อมีการส่ง Session Message ออกไป Method นี้จะถูกเรียกใช้งาน

และการเขียน Application ให้เชื่อมต่อกันระหว่าง Server และ Client มีขั้นตอนดังนี้

ขั้นตอนที่ 1 สร้าง Config File (.cfg) สำหรับนำไปสร้างเป็น Session

Server หรือ Acceptor

```
[DEFAULT]
ConnectionType=acceptor
SocketAcceptPort=6001
SocketReuseAddress=Y
StartTime=00:00:00
EndTime=00:00:00
UseDataDictionary=N
ResetOnLogon=Y
ResetOnLogout=Y
ResetOnDisconnect=Y
FileLogPath=C:\FIXServer\FIXServer\log
FileStorePath=C:\FIXServer\FIXServer\store

[SESSION]
BeginString=FIXT.1.1
DefaultApplVerID=FIX.5.0
SenderCompID=SERVER1
SenderSubID=01
TargetCompID=024
TargetSubID=1
TransportDataDictionary=C:\FIXServer\FIXServer\spec\FIXT11.xml
AppDataDictionary=C:\FIXServer\FIXServer\spec\FIX50.xml
SocketAcceptPort=6001

[SESSION]
BeginString=FIXT.1.1
DefaultApplVerID=FIX.5.0
SenderCompID=SERVER1
SenderSubID=01
TargetCompID=024
TargetSubID=2
TransportDataDictionary=C:\FIXServer\FIXServer\spec\FIXT11.xml
AppDataDictionary=C:\FIXServer\FIXServer\spec\FIX50.xml
SocketAcceptPort=6001
```

Client หรือ Initiator

```
[DEFAULT]
ConnectionType=initiator
ReconnectInterval=60
FileStorePath=store
FileLogPath=log
StartTime=00:00:00
EndTime=00:00:00
UseDataDictionary=N
HttpAcceptPort=9911
SocketConnectHost=127.0.0.1
SocketConnectPort=6001
LogoutTimeout=60
ResetOnLogon=Y
HeartBtInt=60

[SESSION]
BeginString=FIXT.1.1
DefaultApplVerID=FIX.5.0
SenderCompID=024
SenderSubID=1
TargetCompID=SERVER1
TargetSubID=01
FileStorePath=store
SocketConnectHost=127.0.0.1
SocketConnectPort=6001
TransportDataDictionary=spec\FIXT11.xml
```

จาก Config File ด้านบนทั้ง Server และ Client จะเห็นว่ามี Header อยู่ 2 ประเภท คือ

1. [DEFAULT]

ในส่วนนี้ ใช้กำหนดค่า Default ต่าง ๆ เช่น ConnectionType (เป็น Server หรือ Client), SocketAcceptPort (พอร์ทที่ใช้ในการเชื่อมต่อ), FileLogPath (Path ที่ใช้บันทึก Log File) เป็นต้น โดยค่าและรายละเอียดต่างต่าง ๆ สามารถดูได้ที่เว็บไซต์ http://www.quickfixn.org หรือ http://www.quickfixj.org

2. [SESSION]

เป็นส่วนที่ FIX Application นำไปใช้สร้าง Session เพื่อทำการเชื่อมต่อกัน โดยค่าที่ใช้ระบุตัวตนของเครื่องต่าง ๆ คือ BeginString, SenderComID, SenderSubID, TargetComID และ TargetSubID สังเกตุ Config File ของ Server จะเห็นว่าสามารถกำหนด [SEESION] ได้หลายส่วน ซึ่งก็คือสามารถรองรับการเชื่อมต่อจาก Client หลาย ๆ เครื่องตามที่ระบุไว้ใน [SESSION] (จากตัวอย่าง Config File ด้านบนมี 2 ส่วน)

ขั้นตอนที่ 2-1 สร้าง Session จาก Config File (Server)

C#

```
string configFile = @"C:\FIXServer\FIXServer\config\Server.cfg";
SessionSettings sessionSettings = new SessionSettings(configFile);
IApplication application = new ServerApp(sessionSettings);
IMessageStoreFactory storeFactory = new FileStoreFactory(sessionSettings);
ILogFactory screenFactory = new ScreenLogFactory(false, false, false);
ILogFactory logFactory = new FileLogFactory(sessionSettings);
```

Java

```
SessionSettings settings = new SessionSettings(new
    FileInputStream("config/acceptor.cfg"));
FileStoreFactory storeFactory = new FileStoreFactory(settings);
LogFactory logFactory = new FileLogFactory(settings);
MessageFactory messageFactory = new DefaultMessageFactory();
AcceptorApplication acceptorApplication = new AcceptorApplication();
```

ขั้นตอนที่ 2-2 สร้าง Session จาก Config File (Client)

C#

```
string configFile = @"C:\FIXClient\FIXClient\config\Client.cfg";
SessionSettings sessionSettings = new SessionSettings(configFile);
IApplication application = new ClientApp(sessionSettings);
IMessageStoreFactory messageStoreFactory = new
    FileStoreFactory(sessionSettings);
ILogFactory logFactory = new ScreenLogFactory(false, false, false);
```

Java

```
String configFile = "config/initiator.cfg";
SessionSettings sessionSettings = new SessionSettings(new
    FileInputStream(configFile));
FileStoreFactory storeFactory = new FileStoreFactory(sessionSettings);
LogFactory logFactory = new FileLogFactory(sessionSettings);
MessageFactory messageFactory = new DefaultMessageFactory();
InitiatorApp initiatorApp = new InitiatorApp(sessionSettings);
```

ขั้นตอนที่ 3-1 เปิดการเชื่อมต่อ (Server)

C#

```
IAcceptor acceptor = new ThreadedSocketAcceptor(application, storeFactory,
    sessionSettings, logFactory);
```

Java

```
SocketAcceptor acceptor = new SocketAcceptor(acceptorApplication, storeFactory,
    settings, logFactory, messageFactory);
```

ขั้นตอนที่ 3-2 เชื่อมต่อกับ Server (Client)

C#

```
QuickFix.Transport.SocketInitiator initiator =
    new QuickFix.Transport.SocketInitiator(application, messageStoreFactory,
    sessionSettings, logFactory);
```

Java

```
Initiator initiator = new SocketInitiator(initiatorApp, storeFactory,
    sessionSettings, logFactory, messageFactory);
```

ขั้นตอนที่ 4-1 สั่งให้โปรแกรมเริ่มทำงาน (Server)

C#

```
acceptor.Start();
```

Java

```
acceptor.start();
```

ขั้นตอนที่ 4-2 สั่งให้โปรแกรมเริ่มทำงาน (Client)

C#

```
initiator.Start();
```

Java

```
initiator.start();
```

2.12 C# Programming Language (ภาษาซีชาร์ป)

ภาษา C# เป็นภาษาโปรแกรมเชิงวัตถุทำงานบนดอตเน็ตเฟรมเวิร์ก พัฒนาโดยบริษัทไมโครซอฟท์ โดยมีรากฐานมาจากภาษา C++ และภาษาอื่น ๆ โดยเฉพาะภาษาเดลไฟ Delphi และ JAVA

ภาษา C# นั้นเป็นภาษาที่มีรูปร่างหน้าตาและโครงสร้างในแบบที่เรามักจะเรียกว่า “C-Style Language” หรือ ภาษาที่มีลักษณะคล้ายคลึงกับภาษา C นั่นเอง ซึ่งแม้แต่ภาษาที่โปรแกรมเมอร์ชาวไทยคุ้นเคยกันดีอย่าง Java และ PHP นั้นก็จัดอยู่ในภาษากลุ่มนี้เช่นกัน นั่นก็เพราะว่า C-Style เป็นรูปแบบภาษาที่โปรแกรมเมอร์ส่วนใหญ่ที่มักมีพื้นฐานมาจากภาษา C คุ้นเคย แต่ก็อาจจะเป็นภาษาที่ดูแปลกตา สำหรับผู้ที่ไม่มีพื้นฐานการเขียนโปรแกรมมาก่อน หรือผู้ที่คุ้นเคยกับภาษาที่ค่อนข้างดูคล้ายกับภาษาพูดอย่าง Visual Basic ไปเลยก็เป็นได้

ดังนั้น ถ้ามีพื้นฐานจากภาษาในกลุ่ม C-Style อยู่ก่อนแล้ว ก็อาจจะข้ามในส่วนของการแนะนำโครงสร้างภาษานี้ และไปเริ่มอ่านในส่วนของการแนะนำฟีเจอร์เฉพาะของภาษา C# ได้เลย แต่สำหรับผู้ที่เคยพัฒนาโปรแกรมด้วยภาษา Visual Basic มาก่อน หรือผู้ที่กำลังเริ่มเป็นโปรแกรมเมอร์มือใหม่ การแนะนำนี้เป็นส่วนสำคัญ ที่ไม่ควรมองข้าม

2.13 Java Programming Language (ภาษาจาวา)

ภาษาจาวาถูกพัฒนาขึ้นโดยบริษัท Sun Microsystems โดยพัฒนาให้เป็นภาษาที่ใช้ในการเขียนโปรแกรมเชิงวัตถุ (Object-Oriented Programming) และให้สามารถทำงานบนเครื่องคอมพิวเตอร์ที่มีสภาแวดล้อมต่างกันได้โดยไม่ต้อง Compile ไฟล์ใหม่ ซึ่งเรียกคุณสมบัติเช่นนี้ว่า platform Independent และภาษานี้เป็นที่นิยมอย่างแพร่หลายในปี ค.ศ. 1995 เมื่อบริษัท Sun Microsystems ได้นำชุดพัฒนาโปรแกรมภาษาจาวา (Java Development Kit) IDK1.0 ซึ่งเป็นเวอร์ชั่นแรกออกแจกจ่ายบนอินเตอร์เน็ต และหลังจากนั้นก็ได้ออกชุดพัฒนารุ่นต่าง ๆ ตามออกมา จุดเด่นของภาษา Java มีดังนี้

ชุดพัฒนาโปรแกรมสามารถใช้ได้ฟรี เนื่องจากทางผู้พัฒนาภาษา Java ได้ให้ดาวน์โหลดได้ทางอินเตอร์เน็ต

เป็นภาษาที่ออกแบบมาให้ใช้งานง่าย ผู้ที่เคยเขียนโปรแกรมภาษา C มาก้อนสามารถศึกษาได้รวดเร็ว โดยภาษานี้ได้นำไวยากรณ์ส่วนใหญ่ของภาษา C และได้ตัดสิ่งที่ยุ่งยากของภาษา C ออกไป เช่น Pointer เป็นต้น

สามาถนำไปทำงานบนเครื่องคอมพิวเตอร์ที่ต่างระบบกันได้

สนับสนุนการเขียนโปรแกรมเชิงวัตถุ

ทำงานบน Web Browser ได้ โดยมีคุณสมบัติที่เรียกว่า Java Applet ซึ่งทำงานได้บนเว็บถ้าหากโปรแกรม Browser นั้นมีตัวสนับสนุน Java ติดตั้งอยู่

มี Class ต่าง ๆ ให้เรียกใช้จำนวนมาก

สามารถคืนพื้นที่หน่วยความจำได้อัตโนมัติ ในการเขียนโปรแกรมแบบเชิงวัตถุนั้นมักจะต้องมีการจองหน่วยความจำให้กับ Object ต่าง ๆ ภาษา Java จะคืนหน่วยความจำให้อัตโนมัติเมื่อไม่มีการใช้ Object นั้น

2.14 SQL (ภาษาเอสคิวแอล)

SQL ย่อมาจาก **S**tructured **Q**uery **L**anguage คือ ภาษาที่ใช้ในการเขียนโปรแกรม เพื่อจัดการกับฐานข้อมูลโดยเฉพาะ เป็นภาษามาตรฐานบนระบบฐานข้อมูลเชิงสัมพันธ์และเป็นระบบเปิด (open system) หมายถึงเราสามารถใช้คำสั่ง SQL กับฐานข้อมูลชนิดใดก็ได้ และคำสั่งงานเดียวกัน เมื่อสั่งงานผ่านระบบฐานข้อมูลที่แตกต่างกันจะได้ผลลัพธ์เหมือนกัน ทำให้เราสามารถเลือกใช้ฐานข้อมูล ชนิดใดก็ได้โดยไม่ยึดติดกับฐานข้อมูลใดฐานข้อมูลหนึ่ง นอกจากนี้แล้ว SQL ยังเป็นชื่อโปรแกรมฐานข้อมูล ซึ่งโปรแกรม SQL เป็นโปรแกรมฐานข้อมูลที่มีโครงสร้างของภาษา

ที่เข้าใจง่าย ไม่ซับซ้อน มีประสิทธิภาพการทำงานสูง สามารถทำงานที่ซับซ้อนได้โดยใช้คำสั่งเพียงไม่กี่คำสั่ง โปรแกรม SQL จึงเหมาะที่จะใช้กับระบบฐานข้อมูลเชิงสัมพันธ์

ปัจจุบันมีซอฟต์แวร์ระบบจัดการฐานข้อมูล (DBMS) ที่สนับสนุนการใช้คำสั่ง SQL เช่น Oracle, DB2, MS-SQL, MS-Access, MySQL นอกจากนี้ภาษา SQL ถูกนำมาใช้เขียนร่วมกับโปรแกรมภาษาต่างๆ เช่น ภาษา C, C++, C#, Visual Basic, PHP และ Java

2.15 XML (ภาษาเอ็กซ์เอ็มแอล)

โลกของเทคโนโลยีมี Platform มากมายหลากหลายภาษา แต่ละภาษาก็จะคุยกันด้วยสำนวนของมันเอง ในงานที่ต้องใช้ Platform หลากหลาย จะมีปัญหาก็คือมันจะทำงานร่วมกันอย่างไร จะส่งข้อมูลอย่างไร เช่นถ้าจะเขียน ASP.net ให้คุยกับ JSP (Java Server Page) หรือภาษาอื่น ๆ ต้องทำอย่างไร XML จึงเกิดขึ้นมาเพื่อแก้ปัญหานี้ มันจะทำหน้าที่เป็นตัวกลางเก็บข้อมูล โดยที่ ASP.NET, JSP หรือภาษาอื่นๆ ต้องสามารถเข้าใจภาษา XML เป็นมาตรฐานอยู่แล้ว ตัวอย่างเช่น ASP.net ต้องการแลกเปลี่ยนข้อมูลกับ JSP เราก็เขียน ASP.NET ให้สร้างไฟล์ XML พร้อมทั้งใส่โครงสร้างและข้อมูลตามที่เราต้องการ จากนั้นเราก็เขียน JSP ให้มาอ่านไฟล์ XML นั้นๆ แค่นี้ ASP.NET ก็แลกเปลี่ยนข้อมูลกับ JSP ได้แล้ว

XML ย่อมาจาก Extensible Markup Language เป็นภาษาหนึ่งที่ใช้ในการแสดงผลข้อมูล ถ้าเปรียบเทียบกับภาษา HTML จะแตกต่างกันที่ HTML ถูกออกแบบมาเพื่อการแสดงผลอย่างเดียวเท่านั้น เช่นให้แสดงผลตัวเล็ก ตัวหนา ตัวเอียง เหมือนที่คุณเคยเห็นในเวบเพจทั่วไป แต่ภาษา XML นั้นถูกออกแบบมาเพื่อเก็บข้อมูล โดยทั้งข้อมูลและโครงสร้างของข้อมูลนั้นๆไว้ด้วยกัน

ภาษา XML มีโครงสร้างที่ประกอบด้วยแท็ก (Tag) เปิดและแท็กปิด เช่นเดียวกับภาษา HTML แต่ภาษา XML คุณสามารถสร้างแท็กรวมทั้งกำหนดโครงสร้างของข้อมูลได้เอง ซึ่งความสามารถตรงนี้ตัวภาษา HTML ทำไม่ได้เพราะภาษา HTML ถูกกำหนดแท็กตายตัวโดย W3C (The World Wide Web Consortium)

ตัวอย่างที่ 1 XML Tag ที่บอกโครงสร้างและข้อมูลของบุคคล

```xml
<?xml version="1.0" encoding="windows-874"?>
<address_book>
    <person gender="M">
        <name>Jane Doe</name>
        <address>
            <street>123 Main St.</street>
                <city>San Francisco</city>
                    <state>CA</state>
                    <zip>94117</zip>
        </address>
        <phone>555-1212</phone>
    </person>
</address_book>
```

ตัวอย่างที่ 2

```xml
<?xml version="1.0" encoding="windows-874"?>
<callme>
    <my_mobile>0-1307-8072</my_mobile>
    <my_phone>0-2872-8936</my_phone>
</callme>
```

จากตัวอย่างที่ 2 บรรทัดแรก เป็นการประกาศว่าเอกสารนี้เป็นไฟล์ XML นี้มีการเข้ารหัสอักขระแบบ Windows-874 เพื่อปลายทางจะได้เข้าใจและถอดรหัสได้ถูกต้อง จากตัวอย่างจะเห็นว่า จริงๆแล้วในภาษา XML จะแบ่ง โครงสร้างเป็นสองส่วนใหญ่ ๆ คือ Tag และ Element

ตัวอย่างที่ 3 แสดงโครงสร้างของ Tag และ Element

```xml
<root>
    <element>
        <tag></tag>
    </element>
</root>
```

ลงรายละเอียดเกี่ยวกับ Tag กับ Element

Tag สำหรับใน XML มีความหมายในลักษณะเดียวกับที่ใช้ใน HTML Tag คือข้อความที่อยู่ระหว่างสัญลักษณ์ "<" และ ">" มีสองแบบคือ

1. แท็กเปิด (Start tag) เช่น
2. แท็กปิด (End Tag) เช่น สังเกตุได้ว่าแท็กปิดเครื่องหมาย / อยู่หลังสัญลักษณ์ "<"

จากตัวอย่างที่ 2 Tag คือ ....</callme>

Element คือโครงสร้างหลักของ XML ซึ่งอยู่ในรูปของแท็กจะมีลักษณะซ้อนกันเป็นชั้น ๆ โดย Element เริ่มต้นที่แท็กเปิดและสิ้นสุดที่แท็กปิดในแท็กดียวกัน และ Root Element จะเป็น Element บนสุดของไฟล์ XML จากตัวอย่างที่ 2 Element คือ my_mobile, my_phone

Content ข้อมูลที่เก็บใน Element เช่นจากตัวอย่างที่ 2 Content ใน Element my_mobile ก็คือ 0-1307-8072

Attribute คือข้อมูลความหมายเพิ่มเติมเป็นค่าคงที่ ถูกเขียนอยู่ภายใน tag เปิด <....> จะมีมากกว่า 1 หรือมี 1 อันหรือไม่มีเลยก็ได้ จากตัวอย่างที่ 1 ก็คือ Gender="M" เป็นต้น

2.16 Microsoft Visual Studio (ไมโครซอฟท์ วิชวลสตูดิโอ)

Microsoft Visual Studio คือ IDE (Integrated Development Environment) พัฒนาขึ้นโดย Microsoft ซึ่งเป็นเครื่องมือที่ช่วยนักพัฒนาซอฟต์แวร์พัฒนาโปรแกรมคอมพิวเตอร์ Website, Web Application และ Web Service เป็นต้น ในปัจจุบัน Microsoft Visual Studio นั้นสามารถใช้ภาษาโปรแกรมที่เป็นภาษา .NET (ดอทเน็ต) ในโปรแกรมเดียวกัน เช่น VB.NET, C++, C# และ J# เป็นต้น

รูปที่ 2.13 สัญลักษณ์โปรแกรม Microsoft Visual Studio 2010

2.17 Eclipse (อีคลิปส์)

Eclipse คือ โปรแกรม IDE (Integrated Development Environment) สำหรับพัฒนา Applications โดยใช้ JAVA หรือภาษาอื่น ๆ เช่น C, C++, Python, PERL, Ruby ฯลฯ

Eclipse มีการรองรับ Plug-in ที่หลากหลาย ผู้พัฒนาที่ใช้ภาษา Java ในการพัฒนา Applications ต่าง ๆ สามารถใช้ Eclipse ในการพัฒนาได้ โดยตัว Eclipse มีสภาวะแวดล้อมที่สมบูรณ์ คือมีเครื่องมือต่าง ๆ ให้ใช้พร้อม นอกจากนี้ Eclipse ยังสามารถใช้พัฒนาโปรแกรมภาษาอื่น ๆ ได้ถ้ามีตัว Plug-in นั้นอยู่ เช่น ถ้าเราต้องการพัฒนา Applications โดยใช้ภาษา PHP ถ้า Eclipse มีปลั๊กอินสำหรับภาษานี้ เราสามารถใช้ Eclipse ในการพัฒนาได้

รูปที่ 2.14 สัญลักษณ์โปรแกรม Eclipse

2.18 MySQL (มายเอาคิวแอล)

MySQL จัดเป็นระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (RDBMS: Relational Database Management System) ตัวหนึ่ง ซึ่งเป็นที่นิยมกันมากนปัจจุบัน โดยเฉพาะอย่างยิ่งในโลกของอินเตอร์เน็ต สาเหตุเพราะว่า MySQL เป็น ฟรีแวร์ทางด้านฐานข้อมูล さ ที่มีประสิทธิภาพสูง เป็นทางเลือกใหม่จากผลิตภัณฑ์ じ ระบบจัดการฐานข้อมูลในปัจจุบันที่มักจะเป็นการผูกขาดของผลิตภัณฑ์เพียงไม่กี่ตัว นักพัฒนาระบบฐานข้อมูลที่เคยใช้ MySQL ต่างยอมรับในความสามารถ ความรวดเร็ว การรองรับจำนวนผู้ さ ใช้ さ และขนาดของข้อมูลจำนวนมหาศาล ทั้งยังสนับสนุนการใช้งานบนระบบปฏิบัติการมากมายไม่ว่าจะเป็น UNIX, Mac OS, Windows ก็ตาม นอกจากนี้ MySQL ยังสามารถใช้งานร่วมกับ Web Development Platform ทั้งหลายไม่ว่าจะเป็น C, C++, PERL, Python หรือ ASP

MySQL จัดเป็นซอฟต์แวร์ประเภท Open Source Software สามารถดาวน์ じ โหลด Source Code ต้นฉบับได้จากอินเตอร์เน็ตโดยไม ご เสียค่าใช้จ่ายใด ๆ การแก้ไขก็สามารถกระทำได้ さ ตามความต้องการ MySQL ยึดถือสิทธิบัตรตาม GPL (GNU General Public License) ซึ่งเป็นข้อกำหนดของซอฟต์แวร์ประเภทนี้ส่วนใหญ่ ご โดยจะเป็นการชี้แจงว่า สิ่งใดทำได้ さ หรือทำไม ご ได้สำหรับการใช้ さ งานในกรณีต่าง ๆ ทั้งนี้ถ้าต้องการข้อมูลเพิ่มเติมหรือรายละเอียดของ GPL สามารถหาข้อมูลได้จากเว็บไซต์ http://www.gnu.org/

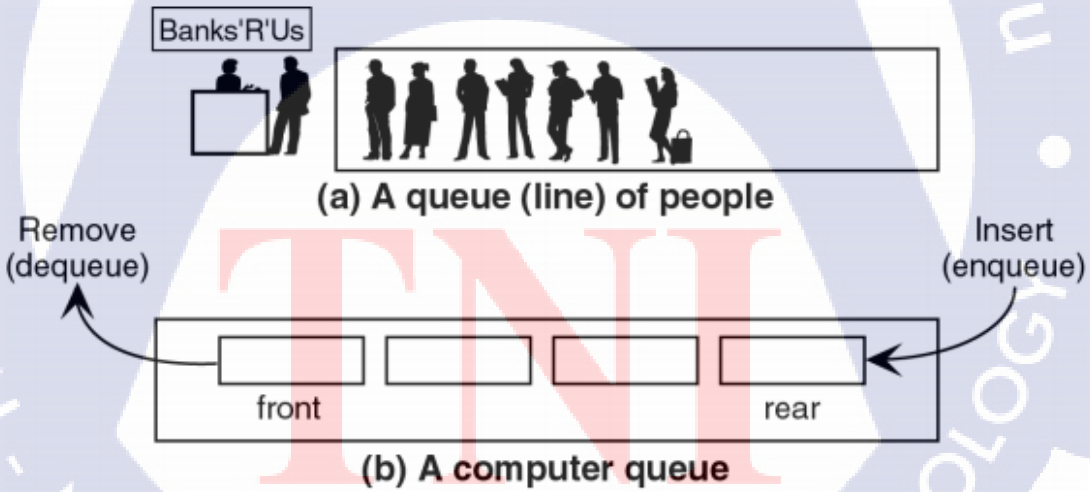
ทุกวันนี้มีการนำ MySQL ไปใช้ในระบบต่าง ๆ มากมายไม่ว่าจะเป็นระบบเล็ก ๆ ที่มีจำนวนตารางข้อมูลน้อย มีความสัมพันธ์ของข้อมูลในแต่ละตารางไม่ซับซ้อน เช่น ระบบฐานข้อมูลบุคคลในแผนกเล็ก ๆ ไปจนถึงระบบจัดการข้อมูลขนาดใหญ่ที่ประกอบด้วยตารางข้อมูลมากมาย มีความสัมพันธ์ของข้อมูลในแต่ละตารางซับซ้อน เช่น ระบบสต็อกสินค้า ระบบบัญชีเงินเดือน เป็น

ต้นโดยเฉพาะอย่างยิ่งในปัจจุบันมีการใช้ MySQL เป็น Database Server เพื่อการทำงานสำหรับ Web Database Application ในโลกของอินเตอร์เน็ตมากขึ้น

รูปที่ 2.15 สัญลักษณ์โปรแกรม MySQL

2.19 Queue (คิว)

คิวเป็นโครงสร้างข้อมูลพื้นฐานที่มีการจำกัดการเพิ่มข้อมูลและการเอาข้อมูลออก เช่นเดียวกับแสตก แต่ลักษณะของคิวก็คือข้อมูลใดที่มาก่อนจะถูกดำเนินการก่อน (First-in First-out : FIFO) คิวเป็นโครงสร้างข้อมูลที่เป็นแถวลำดับแบบอันดับซึ่งการเพิ่มข้อมูลจะกระทำที่ปลายด้านหนึ่งเรียกว่า “Rear” (หางคิว) โดยโอเปอร์เรชั่น Enqueue และการนำเอาข้อมูลออกจากคิวจะกระทำที่ปลายอีกด้านหนึ่งเรียกว่า “Front” (หัวคิว) โดยโอเปอเรชัน Dequeue ดังรูปที่ 2.16



รูปที่ 2.16 รูปแบบของคิว

บทที่ 3
แผนงานการปฏิบัติงานและขั้นตอนการปฏิบัติงาน

3.1 แผนงานการปฏิบัติงาน

แผนงาน	ระยะเวลา (เดือน)															
	มิถุนายน				กรกฎาคม				สิงหาคม				กันยายน			
1. ศึกษาเทคโนโลยีและทฤษฎีที่เกี่ยวข้องเกี่ยวกับ FIX Protocol																
2. ออกแบบการทำงานของโปรแกรม																
4. จัดทำโปรแกรม																
5. ทดสอบและประเมินผลการทำงานของโปรแกรม																
6. จัดทำเอกสารประกอบโครงงาน																

ตารางที่ 3.1 ตารางแสดงแผนการปฏิบัติงาน

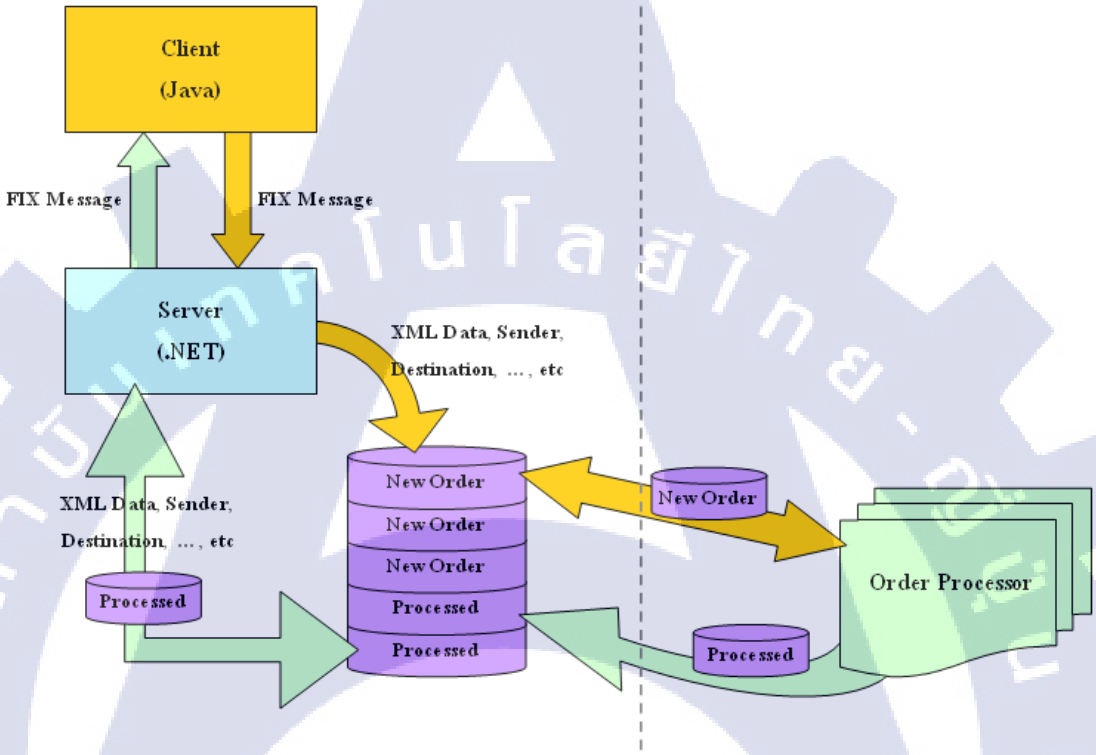
3.2 รายละเอียดโครงงาน

ปัจจุบัน ระบบ Transmit Message ในตลาดหลักทรัพย์ซึ่งใช้ MQ และ BizTalk เป็นตัวกลาง (Middleware) ในการจัดการการรับ-ส่ง Message ต่าง ๆ มีค่าใช้จ่ายในการดำเนินงานค่อนข้างสูง เพื่อลดค่าใช้จ่ายในการดำเนินงานส่วนนี้ลง จึงได้นำซอฟแวร์และเทคโนโลยีอื่น ๆ ที่มีความสามารถเทียบเท่าหรือมีประสิทธิภาพมากกว่ามาใช้ทดแทนระบบเดิมซึ่งซอฟต์แวร์ที่เลือกมาใช้ทำการทดสอบนี้ก็คือ FIX Engine (QuickFIX) กับ MySQL ทั้งสองซอฟต์แวร์นี้เป็น Open Source Software ที่สามารถ Download มาพัฒนาต่อยอดได้

ในโครงงานได้เขียนโปรแกรมให้ Server ซึ่งเป็น .NET Platform กับ Client ซึ่งเป็น Java Platform ทำการรับ-ส่ง FIX Message กันโดยใช้ FIX Engine ร่วมกับ MySQL เพื่อทดสอบว่าการ

จัดการ Message ด้วยการใช้ซอฟต์แวร์ทั้งสองดังกล่าวนั้นสามารถทำงานร่วมกันได้หรือไม่ และถ้าหากจะนำมาใช้ทดแทนจะมีประสิทธิภาพเพียงพอที่จะทดแทนระบบเดิมได้หรือเปล่า

การเขียน Application เพื่อจัดการการรับ-ส่ง Message ได้ออกแบบการทำงานไว้ดังรูปที่ 3.1



รูปที่ 3. 1 แผนภาพแสดงการรับ-ส่ง FIX Message ที่ออกแบบสำหรับนำไปเขียนโปรแกรม

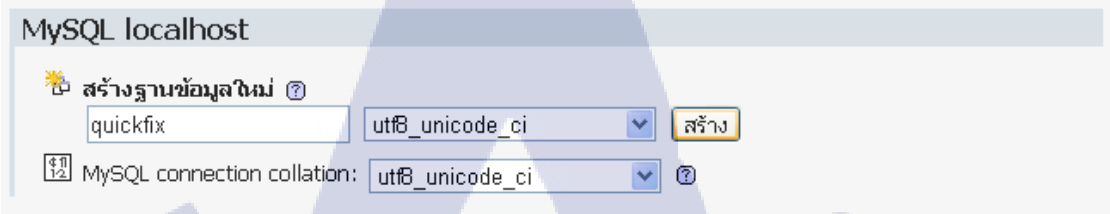
ในระบบนี้ประกอบด้วยฝั่ง Client ที่เป็น Java Platform และฝั่ง Server ที่เป็น .NET Platform ในส่วนของ Client จะมีหน้าที่ส่ง Order ต่าง ๆ ในรูปแบบ FIX Message มาที่ Server จากนั้น Server จะแกะข้อมูลจาก FIX Message นำมาเก็บลงในฐานข้อมูล โดยเก็บเป็น XML เพื่อให้ Order Processor นำข้อมูลไปประมวลผลต่อไป

ในส่วนของการส่ง FIX Message ที่ต้องส่งจาก Server ไปยัง Client นั้นได้เขียนโปรแกรมให้ Server Select Record ที่รอการถูกส่งในฐานข้อมูล และวิธีการ Select Record มาสร้างเป็น FIX Message แล้วส่งไปให้ Client แต่ละครั้งจะใช้หลักการของคิวในการจัดการ

3.3 ขั้นตอนการดำเนินงานโครงงาน

1. เตรียมฐานข้อมูล

1. เปิดโปรแกรม MySQL และสร้างฐานข้อมูลชื่อว่า quickfix ดังรูปที่ 3.2



รูปที่ 3.2 สร้างฐานข้อมูลในโปรแกรม MySQL

2. สร้าง Table fix_message และ session โดยโครงสร้าง ดังรูปที่ 3.3 และ 3.4

ฟิลด์	ชนิด	การเรียงลำดับ	แอตทริบิวต์	ว่างเปล่า (null)	ค่าปริยาย	เพิ่มเติม
id	int(50)			ไม่	None	AUTO_INCREMENT
beginstring	varchar(10)	utf8_unicode_ci		ไม่	None	
msgtype	varchar(2)	utf8_unicode_ci		ไม่	None	
sendercomid	varchar(50)	utf8_unicode_ci		ไม่	None	
sendersubid	varchar(50)	utf8_unicode_ci		ไม่	None	
targetcomid	varchar(50)	utf8_unicode_ci		ไม่	None	
targetsubid	varchar(50)	utf8_unicode_ci		ไม่	None	
xmldata	text	utf8_unicode_ci		ไม่	None	
inboundmsg	text	utf8_unicode_ci		ไม่	None	
outboundmsg	text	utf8_unicode_ci		ไม่	None	
status	int(1)			ไม่	None	

รูปที่ 3.3 โครงสร้าง Table fix_message

ฟิลด์	ชนิด	การเรียงลำดับ	แอตทริบิวต์	ว่างเปล่า (null)	ค่าปริยาย	เพิ่มเติม
id	int(11)			ไม่	None	AUTO_INCREMENT
beginstring	varchar(10)	utf8_unicode_ci		ไม่	None	
sendercomid	varchar(50)	utf8_unicode_ci		ไม่	None	
sendersubid	varchar(50)	utf8_unicode_ci		ไม่	None	
targetcomid	varchar(50)	utf8_unicode_ci		ไม่	None	
targetsubid	varchar(50)	utf8_unicode_ci		ไม่	None	

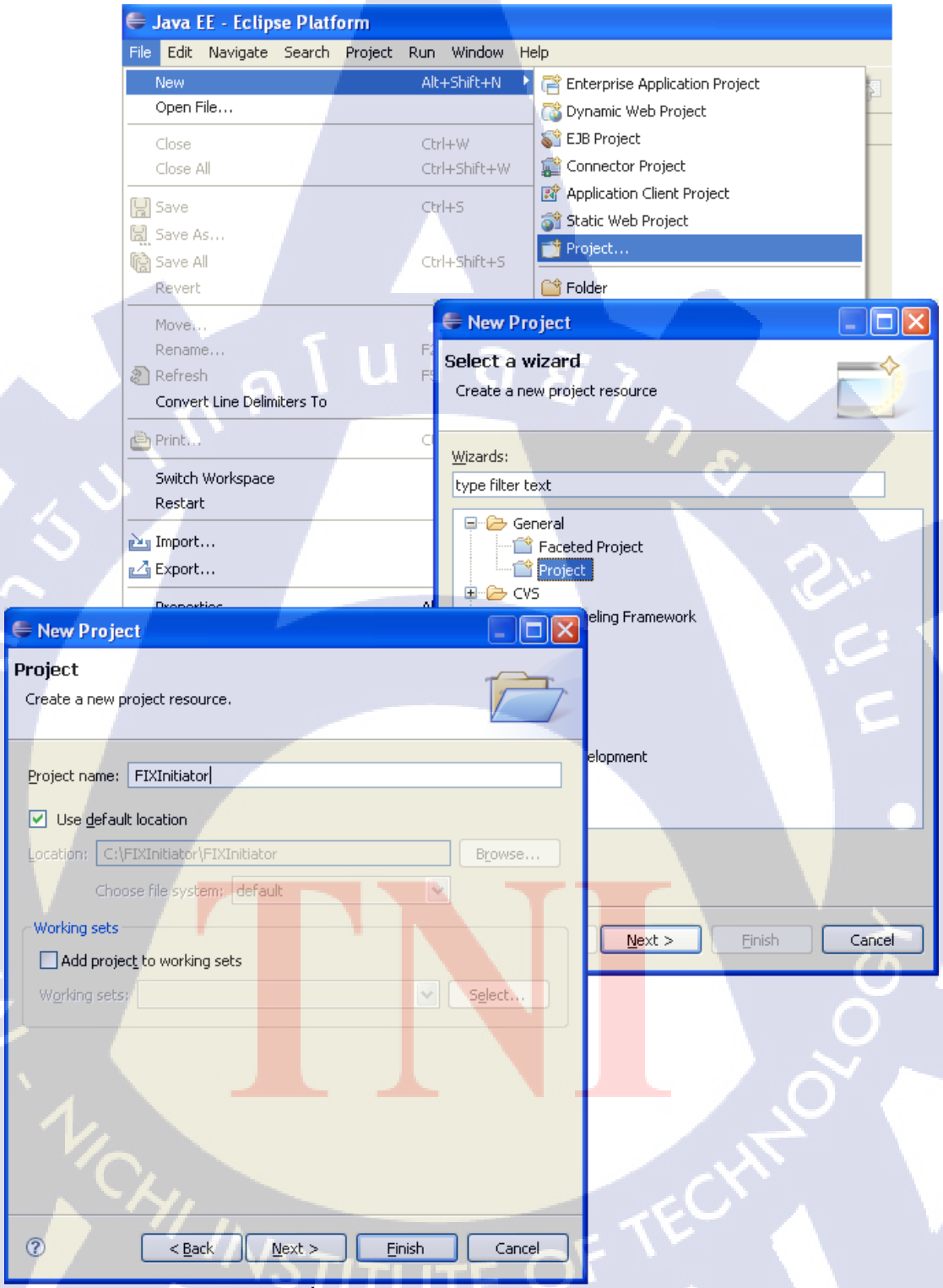
รูปที่ 3.4 โครงสร้าง Table session

2. เขียนโปรแกรม FIX Application ของ Client ด้วย Java

1. สร้าง Config File ตามตัวอย่างโค้ดด้านล่าง

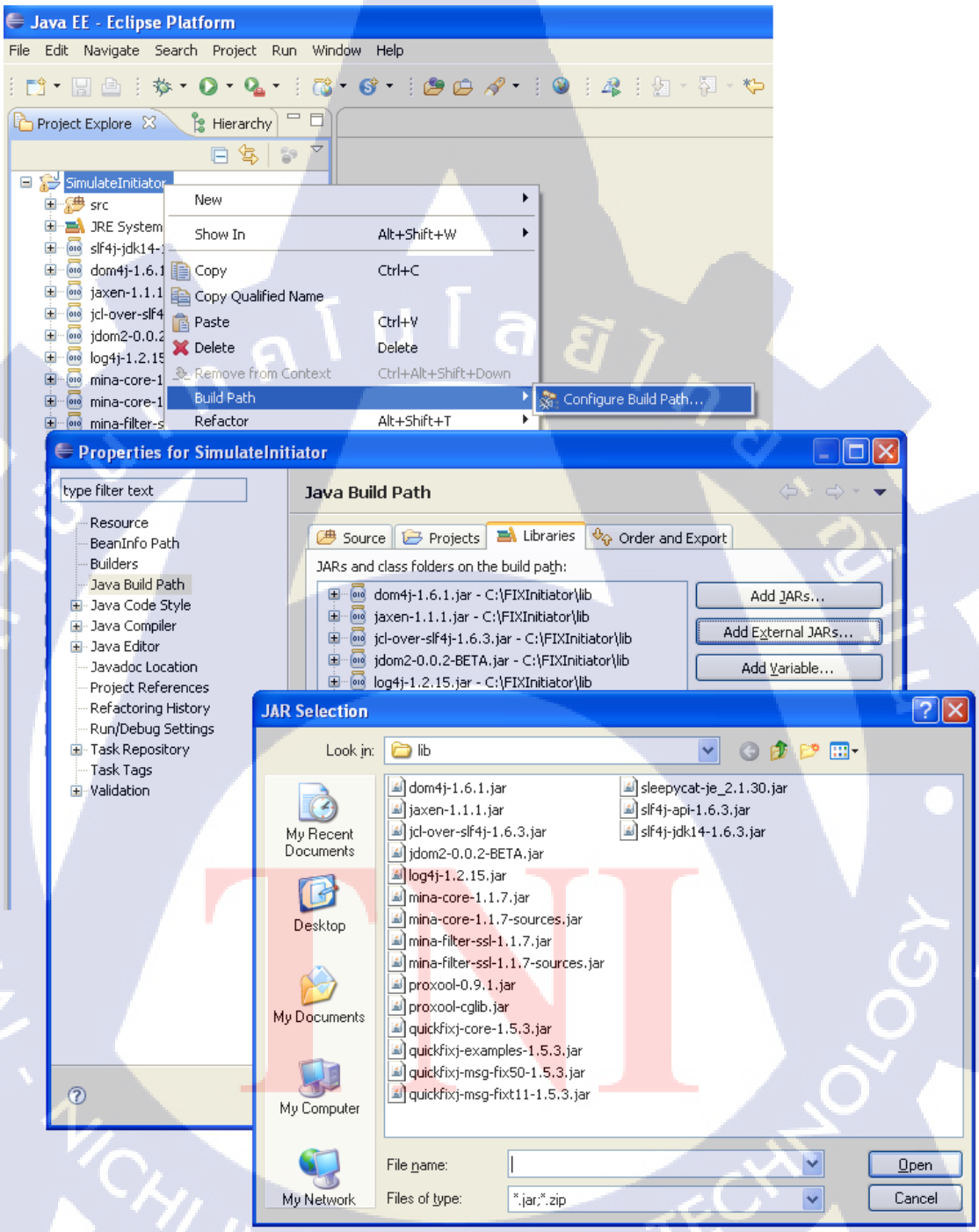
```
[DEFAULT]
ConnectionType=initiator
ReconnectInterval=60
FileStorePath=store
FileLogPath=log
StartTime=00:00:00
EndTime=00:00:00
UseDataDictionary=N
HttpAcceptPort=9911
SocketConnectHost=127.0.0.1
SocketConnectPort=6001
LogoutTimeout=60
ResetOnLogon=Y
HeartBtInt=60

[SESSION]
BeginString=FIXT.1.1
DefaultApplVerID=FIX.5.0
SenderCompID=024
SenderSubID=1
TargetCompID=SERVER1
TargetSubID=01
FileStorePath=store
SocketConnectHost=127.0.0.1
SocketConnectPort=6001
TransportDataDictionary=spec\FIXT11.xml
```

2. เปิดโปรแกรม Eclipse และ New Project ดังวิธีในรูปที่ 3.5



รูปที่ 3.5 วิธี New Project ในโปรแกรม Eclipse

2. ทำการ Add External JARs ที่เกี่ยวข้องในการเขียนโปรแกรม FIX Application บน Java ดังวิธีในรูปที่ 3.6 โดยไฟล์ต่าง ๆ นั้นสามารถดาวน์โหลดได้จากเว็บไซต์ http://www.quickfixj.org



รูปที่ 3.6 วิธี Add External JARs สำหรับเขียนโปรแกรม FIX Application

3. ทำการ Import Package เข้ามาใน Project ดังโค้ดด้านล่าง

```
1  import java.io.*;
2  import quickfix.*;
```

บรรทัดที่ 1 ใช้สำหรับอ่านข้อมูลใน Config File

บรรทัดที่ 2 ใช้สำหรับเขียนโปรแกรม FIX Application

3. สร้าง Session และเชื่อมต่อกับ FIX Engine ของ Server ดังโค้ดด้านล่าง

```
1  import quickfix.*;
2  public static void main(String[] args)
3  {
4      try
5      {
6          String configFile = "config/initiator.cfg";
7          SessionSettings sessionSettings = new SessionSettings(new
8              FileInputStream(configFile));
9          FileStoreFactory storeFactory = new FileStoreFactory(sessionSettings);
10         LogFactory logFactory = new FileLogFactory(sessionSettings);
11         MessageFactory messageFactory = new DefaultMessageFactory();
12         InitiatorApp initiatorApp = new InitiatorApp(sessionSettings);
13         Initiator initiator = new SocketInitiator(initiatorApp, storeFactory,
14             sessionSettings, logFactory, messageFactory);
15         initiator.start();
16         System.in.read();
17         initiator.stop();
18     }
19     catch (Exception e)
20     {
21         System.out.println(e.toString());
22     }
23 }
```

4. สร้าง Class สำหรับเขียนโปรแกรมทดสอบการส่ง FIX Message ไปที่ Server ในที่นี้ตั้งชื่อคลาสว่า InitiatorApp ดังโค้ดด้านล่าง

```
1  import quickfix.*;
2  public class InitiatorApp implements Application
3  {
4      private SessionID sessionID = null;
5      private SessionSettings sessionSettings = null;
6      public InitiatorApp(SessionSettings sessionSettings)
7      {
8          this.sessionSettings = sessionSettings;
9      }
10     @Override
11     public void fromAdmin(Message message, SessionID sessionID){ }
12     @Override
13     public void fromApp(Message message, SessionID sessionID) { }
14     @Override
15     public void onCreate(SessionID sessionID)
16     {
17         this.sessionID = sessionID;
18     }
19     @Override
20     public void onLogon(SessionID sessionID) { }
21     @Override
22     public void onLogout(SessionID sessionID) { }
23     @Override
24     public void toAdmin(Message message, SessionID sessionID) { }
25     @Override
26     public void toApp(Message message, SessionID sessionId) { }
27 }
```

บรรทัดที่ 1 ทำการ Import Package ในการเขียน FIX Application

บรรทัดที่ 2 ทำการ Implement Interface Application

บรรทัดที่ 10-26 โปรแกรมจะกำหนดให้ต้องเขียนโค้ดการทำงานเพิ่มเติมใน Method fromAdmin, fromApp, onCreate, onLogon, onLogout, toAdmin และ toApp ตามข้อกำหนดที่อยู่ใน Interface Application โดยการทำงานของแต่ละ Method จะถูกเรียกใช้งานเมื่อไหร่นั้นได้อธิบายไว้แล้วในบทที่ 2

5. เขียนโค้ดให้ทำการส่ง FIX Message ไปที่ FIX Engine ของ Server ดังโค้ดตัวอย่างด้านล่าง

```
1  @Override
2  public void onLogon(SessionID sessionID)
3  {
4      System.out.println("On logon : " + sessionID);
5      sendFIXMessage();
6  }
7  public void sendFIXMessage()
8  {
9      try
10     {
11         Message message = new Message();
12         message.getHeader().setField(new quickfix.field.MsgType("n"));
13         String xmlData = "<request><msgcd>1111</msgcd>" +
14             "<rqid>2222</rqid></request>";
15         message.setInt(212, xmlData.length());
16         message.setString(213, xmlData);
17         Session.sendToTarget(message, sessionID);
18     }
19     catch (SessionNotFound e)
20     {
21         System.out.print(e.toString());
22     }
23 }
```

บรรทัดที่ 4 -5 หลังจากที่มีการ Logon สำเร็จ โปรแกรมจะพิมพ์ค่า SessionID ออกมาแสดงทาง Console และเรียกใช้งาน Method sendFIXMessage

บรรทัดที่ 11-12 สร้าง FIX Message และกำหนด MsgType (แท็ก 35) เป็น n ซึ่งหมายถึงส่ง FIX Message ในรูปแบบ XML non FIX

บรรทัดที่ 13-16 กำหนด XmlDataLen (แท็ก 212) และ XmlData (แท็ก 213) ซึ่งเป็น Required Field สำหรับการส่ง MsgType n

บรรทัดที่ 17 ส่ง FIX Message ไปที่ FIX Engine ของ Server

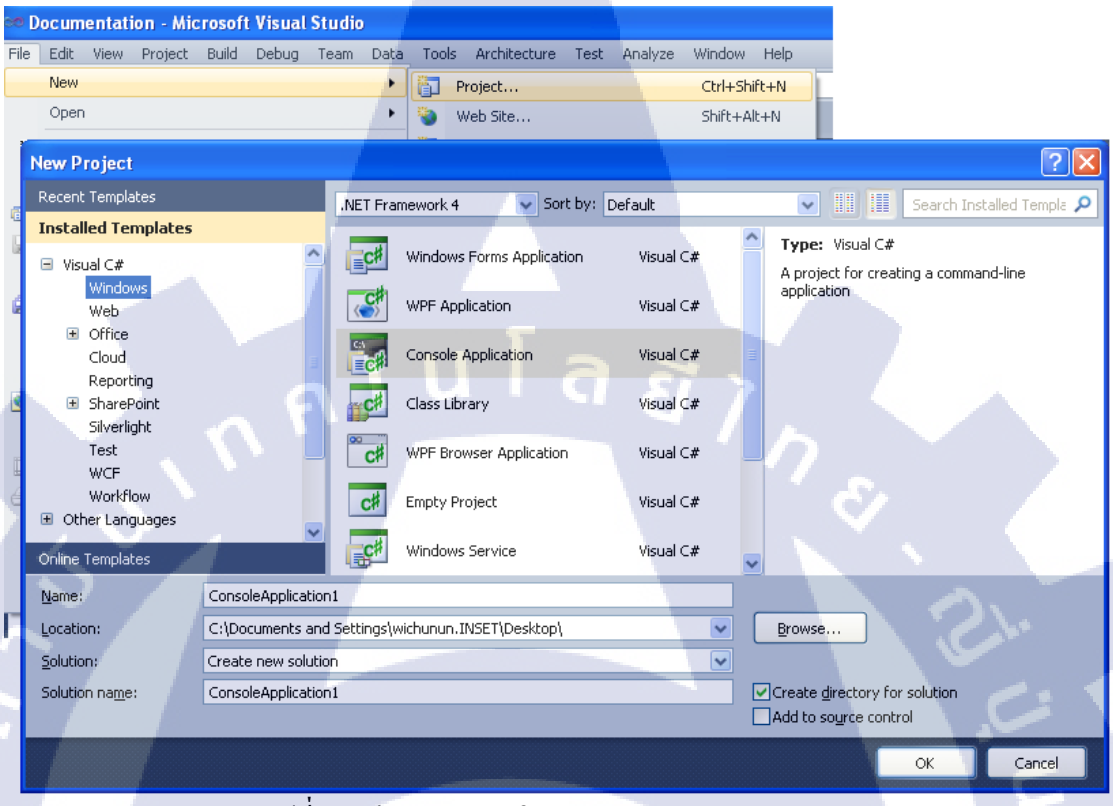
3. เขียนโปรแกรม FIX Engine ของ Server ด้วย C#

1. สร้าง Config File ดังตัวอย่างโค้ดด้านล่าง

```
[DEFAULT]
ConnectionType=acceptor
SocketAcceptPort=6001
SocketReuseAddress=Y
StartTime=00:00:00
EndTime=00:00:00
UseDataDictionary=N
ResetOnLogon=Y
ResetOnLogout=Y
ResetOnDisconnect=Y
FileLogPath=C:\FIXServer\FIXServer\log
FileStorePath=C:\FIXServer\FIXServer\store

[SESSION]
BeginString=FIXT.1.1
DefaultApplVerID=FIX.5.0
SenderCompID=SERVER1
SenderSubID=01
TargetCompID=024
TargetSubID=1
TransportDataDictionary=C:\FIXServer\FIXServer\spec\FIXT11.xml
AppDataDictionary=C:\FIXServer\FIXServer\spec\FIX50.xml
SocketAcceptPort=6001

[SESSION]
BeginString=FIXT.1.1
DefaultApplVerID=FIX.5.0
SenderCompID=SERVER1
SenderSubID=01
TargetCompID=024
TargetSubID=2
TransportDataDictionary=C:\FIXServer\FIXServer\spec\FIXT11.xml
AppDataDictionary=C:\FIXServer\FIXServer\spec\FIX50.xml
SocketAcceptPort=6001
```

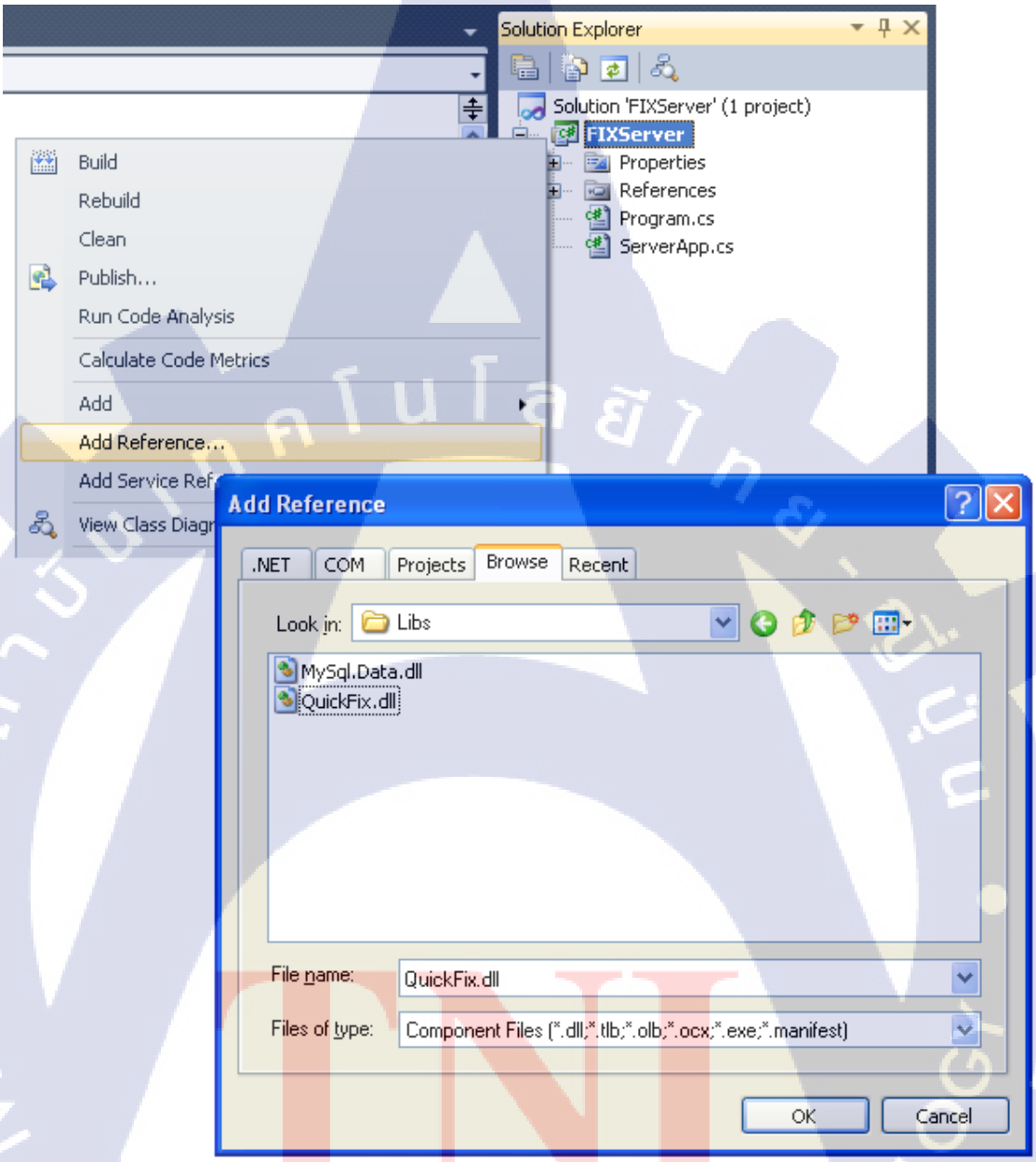
Config File ของ Server ในที่นี้ได้กำหนดให้สามารถเชื่อมต่อกับClient ในลักษณะ One to Many สังเกตุได้จาก Heading [SESSION] ที่มีสองส่วน ซึ่งก็คือ Server จะเปิดรับการเชื่อมต่อจาก Client ได้สอง Session

2. เปิดโปรแกรม Microsoft Visual Studio แล้วทำการ New Project โดยเลือกชนิดของ Project เป็นแบบ Console Application ในที่นี้ตั้งชื่อ Project ว่า FIXServer ดังขั้นตอนในรูปที่ 3.7



รูปที่ 3.7 วิธี New Project ใน Microsoft Visual Studio 2010

3. ทำการ Add References โดยเพิ่มไฟล์ .DLL ที่ใช้สำหรับการเขียน FIX Application บน .NET ดังขั้นตอนในรูปที่ 3.8



รูปที่ 3.8 วิธี Add Reference ใน Microsoft Visual Studio 2010

Mysql.Data.dll ใช้สำหรับเชื่อมต่อฐานข้อมูล MySQL สามารถดาวน์โหลดได้จาก http://dev.mysql.com/downloads/connector/net/

QuickFIX.dll ใช้สำหรับเขียนโปรแกรม FIX Application บน .NET สามารถดาวน์โหลดได้จาก http://www.quickfixj.org

4 . เขียนโค้ดสร้าง Session และเปิดรับการเชื่อมต่อจาก Client ในไฟล์ Program.cs ที่โปรแกรม Microsoft Visual Studio สร้างไว้ให้ โดยเขียนไว้ใน Method Main ดังโค้ดด้านล่าง

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using QuickFix;

namespace FIXServer
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Press <enter> to quit.");
            string configFile = @"C:\FIXServer\FIXServer\config\Server.cfg";
            SessionSettings sessionSettings = new SessionSettings(configFile);
            IApplication application = new ServerApp(sessionSettings);
            IMessageStoreFactory storeFactory = new FileStoreFactory(
                sessionSettings);
            ILogFactory screenFactory = new ScreenLogFactory(false, false,
                false);
            ILogFactory logFactory = new FileLogFactory(sessionSettings);
            IAcceptor server = new ThreadedSocketAcceptor(application,
                storeFactory, sessionSettings, logFactory);
            server.Start();
            Console.ReadKey();
            server.Stop();
        }
    }
}
```

2. สร้าง Class สำหรับเขียนการทำงานตามที่ได้ออกแบบไว้ ในที่นี้ตั้งชื่อคลาสว่า ServerApp ดังโค้ดด้านล่าง

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using QuickFix;
6  using QuickFix.Fields;
7  using MySql.Data.MySqlClient;
8  using System.Timers;
9  using System.Data;
10 namespace FIXServer
11 {
12     class ServerApp : MessageCracker, IApplication
13     {
14         private SessionSettings settings;
15         private SessionID sessionID;
16         public ServerApp(SessionSettings settings)
17         {
18             this.settings = settings;
19         }
20         public void OnCreate(SessionID sessionID)
21         {
22             this.sessionID = sessionID;
23         }
24         public void OnLogon(SessionID sessionID) { }
25         public void OnLogout(SessionID sessionID) { }
26         public void ToAdmin(Message message, SessionID sessionID) { }
27         public void FromAdmin(Message message, SessionID sessionID) { }
28         public void ToApp(Message message, SessionID sessionId) { }
29         public void FromApp(Message message, SessionID sessionID){ }
30     }
31 }
```

บรรทัดที่ 5-6 ทำการเรียกใช้ Namespace ที่เกี่ยวข้องสำหรับเขียนโปรแกรม FIX Application และ สร้าง FIX Message

บรรทัดที่ 7 ใช้ติดต่อและจัดการข้อมูลในโปรแกรม MySQL

บรรทัดที่ 8 ใช้กำหนด Task Scheduler ในโปรแกรมซึ่งจะอธิบายในข้อถัดไป

บรรทัดที่ 9 ใช้สำหรับนำข้อมูลที่ได้จากการ Query ใน MySQL มาเก็บในรูปแบบ DataSet

บรรทัดที่ 12 ทำการ Implement Interface IApplication

บรรทัดที่ 20-29 โปรแกรมจะกำหนดให้ต้องเขียนโค้ดให้กับ Method FromAdmin, FromApp, OnCreate, OnLogon, OnLogout, ToAdmin และ ToApp ตามข้อกำหนดที่อยู่ใน Interface IApplication โดยการทำงานของแต่ละ Method จะถูกเรียกใช้งานเมื่อไหร่นั้นได้อธิบายไว้แล้วในบทที่ 2

3. ประกาศตัวแปรดังโค้ดด้านล่าง เพื่อนำไปใช้งานใน Method ต่าง ๆ ในโปรแกรม

```
1  string msgType, beginString, senderComID, senderSubID, targetComID,
2      targetSubID, xmlData, inboundMsg, outboundMsg;
3  string connectionStr = "SERVER=localhost;" +
4      "DATABASE=quickfix;" +
5      "UID=wichunun;" +
6      "PASSWORD=123456;";
```

บรรทัดที่ 3-6 ใช้สำหรับ Logon และเชื่อมต่อกับ MySQL

4. ใน Method OnLogon เขียนโปรแกรมให้นำค่า SessionID ของ Client ที่ทำการ Logon สำเร็จมา Insert ลง Table session

```
1  public void OnLogon(SessionID sessionID)
2  {
3      beginString = sessionID.BeginString;
4      senderComID = sessionID.SenderCompID;
5      senderSubID = sessionID.SenderSubID;
6      targetComID = sessionID.TargetCompID;
7      targetSubID = sessionID.TargetSubID;
8      MySqlConnection connection = new MySqlConnection(connectionStr);
9      connection.Open();
10     string sqlStr = "INSERT INTO session(beginstring, sendercomid,
11         sendersubid,";
12     sqlStr += " targetcomid, targetsubid)";
13     sqlStr += " VALUES('" + beginString + "', '" + senderComID + "',";
14     sqlStr += " '" + senderSubID + "', '" + targetComID + "', '" +
15         targetSubID + "')";
16     MySqlCommand sqlCmd = new MySqlCommand(sqlStr, connection);
17     sqlCmd.ExecuteNonQuery();
18     Console.WriteLine("On Logon : " + sessionID);
19 }
```

บรรทัดที่ 3-7 แกะค่า SeesionID มาเก็บไว้ในตัวแปร

บรรทัดที่ 8-17 Insert SessionID ลงฐานข้อมูล เพื่อเป็นการบอกว่าในขณะนั้น มี Client ไหนบ้างที่กำลังออนไลน์อยู่

บรรทัดที่ 18 พิมพ์ค่า SessionID หลัง Logon สำเร็จ

5 . ใน Method OnLogout เขียนโปรแกรมให้ Delete Record ใน Table session ที่ทำการ Logout ไป

```
public void OnLogout(SessionID sessionID)
{
    beginString = sessionID.BeginString;
    senderComID = sessionID.SenderCompID;
    senderSubID = sessionID.SenderSubID;
    targetComID = sessionID.TargetCompID;
    targetSubID = sessionID.TargetSubID;
    MySqlConnection connection = new MySqlConnection(connectionStr);
    connection.Open();
    string sqlStr = "DELETE FROM session WHERE sendercomid = '" +
        senderComID + "' AND";
    sqlStr += " sendersubid = '" + senderSubID + "' AND";
    sqlStr += " targetcomid = '" + targetComID + "' AND";
    sqlStr += " targetsubid = '" + targetSubID + "'";
    MySqlCommand sqlCmd = new MySqlCommand(sqlStr, connection);
    sqlCmd.ExecuteNonQuery();
    Console.WriteLine("On logout : " + sessionID);
}
```

การทำงานจะเหมือนกับใน Method OnLogon เพียงแต่เปลี่ยนจากการ Insert เป็น Delete Client ที่ Logout จาก Server ไปแล้ว

6. ใน Method FromAdmin เขียนโปรแกรมรับ FIX Message จาก Client และนำ FIX Message นั้น ๆ มาแกะข้อมูลใน Fields ต่าง ๆ ก่อนที่จะทำการ Insert ลงในฐานข้อมูล

```
1  public void FromAdmin(Message message, SessionID sessionID)
2  {
3      msgType = message.Header.GetString(35);
4      if (msgType.Equals(MsgType.XML_NON_FIX))
5      {
6          beginString = message.Header.GetField(8);
7          senderComID = message.Header.GetField(49);
8          senderSubID = message.Header.GetField(50);
9          targetComID = message.Header.GetField(56);
10         targetSubID = message.Header.GetField(57);
11         xmlData = message.Header.GetField(213);
12         MySqlConnection connection = new MySqlConnection(connectionStr);
13         connection.Open();
14         string sqlStr = "INSERT INTO fix_message(beginstring, msgtype,";
15             sqlStr += " sendercomid, sendersubid, targetcomid, targetsubid,
16                 xmldata, inboundmsg)";
17             sqlStr += " VALUES('" + beginString + "', '" + msgType + "', '" +
18                 senderComID + "',";
19             sqlStr += " '" + senderSubID + "', '" + targetComID + "', '" +
20                 targetSubID + "',";
21             sqlStr += " '" + xmlData + "', '" + message.ToString().Replace(
22                 (char)1, '^') + "')";
23         MySqlCommand sqlCmd = new MySqlCommand(sqlStr, connection);
24         sqlCmd.ExecuteNonQuery();
25     }
26     Console.WriteLine("From admin : " + sessionID.TargetCompID + "/" +
27         sessionID.TargetSubID + "\n" + message);
28 }
```

บรรทัดที่ 6-10 นำค่าที่แกะออกมาจาก Fields ใน FIX Message มาเก็บใน Variable โดยฟิลด์ที่แกะออกมามีดังนี้ BeginString (แท็ก 8), SenderComID (แท็ก 49), SenderSubID (แท็ก 50), TargetComID (แท็ก 56), TargetSubID (แท็ก 57), XmlData (แท็ก 213)

บรรทัดที่ 12-24 เชื่อมต่อและกำหนดคำสั่ง SQL ที่จะส่งไป Query ที่ MySQL

4. สร้าง Task Scheduled เพื่อนำ Record ในฐานข้อมูลที่มีสถานะ Processed ส่งไปให้ Client

1. สร้าง Method TaskSheduler สำหรับกำหนด Event การทำงาน ดังตัวอย่างโค้ดเนล่าง

```
1  public void TaskScheduler()
2  {
3      Timer timer = new Timer { Interval = 10000, Enabled = true };
4      timer.Elapsed += new ElapsedEventHandler(SendMessage);
5  }
```

บรรทัดที่ 3-4 กำหนดให้ทุก ๆ 10 วินาที โปรแกรมจะเรียกใช้งาน Method SendMessage

2. สร้าง Method SendMessage และเขียนโค้ดการทำงาน ดังโค้ดด้านล่าง

```
public void SendMessage(object sender, ElapsedEventArgs e)
{
    MySqlConnection connection = new MySqlConnection(connectionStr);
    connection.Open();
    string sqlStr = "SELECT id, beginstring, msgtype, sendercomid,
        sendersubid,";
    sqlStr += " targetcomid, targetsubid, outboundmsg";
    sqlStr += " FROM fix_message";
    sqlStr += " WHERE outboundmsg != '' AND status =" + 0;
    sqlStr += " ORDER BY id";
    sqlStr += " LIMIT 0, 1";
    MySqlCommand sqlCmd = new MySqlCommand(sqlStr, connection);
    MySqlDataAdapter adapter = new MySqlDataAdapter(sqlCmd);
    DataSet dataSet = new DataSet();
    adapter.Fill(dataSet, "Queue");
    if ((int)dataSet.Tables["Queue"].Rows.Count > 0)
    {
        beginString =
            dataSet.Tables["Queue"].Rows[0]["beginstring"].ToString();
        senderComID =
            dataSet.Tables["Queue"].Rows[0]["targetcomid"].ToString();
        senderSubID = dataSet.Tables["Queue"].Rows[0][6].ToString();
        targetComID = dataSet.Tables["Queue"].Rows[0][3].ToString();
        targetSubID = dataSet.Tables["Queue"].Rows[0][4].ToString();
        outboundMsg = dataSet.Tables["Queue"].Rows[0][7].ToString();
        QuickFix.Message fixml = new QuickFix.Message();
        MsgType msgType = new MsgType("n");
        fixml.Header.SetField(msgType);
        xmlData = outboundMsg;
        fixml.Header.SetField(new XmlDataLen(xmlData.Length)); // 212
        fixml.Header.SetField(new XmlData(xmlData)); // 213
        SessionID ssID = new SessionID(beginString, senderComID, senderSubID,
            targetComID, targetSubID);
        string sqlStr2 = "SELECT * FROM session";
        sqlStr2 += " WHERE beginstring = '" + beginString + "' AND";
        sqlStr2 += " sendercomid = '" + senderComID + "' AND";
        sqlStr2 += " sendersubid = '" + senderSubID + "' AND";
        sqlStr2 += " targetcomid = '" + targetComID + "' AND";
        sqlStr2 += " targetsubid = '" + targetSubID + "'";
        MySqlCommand sqlCmd2 = new MySqlCommand(sqlStr2, connection);
        MySqlDataAdapter adapter2 = new MySqlDataAdapter(sqlCmd2);
        DataSet dataSet2 = new DataSet();
        adapter2.Fill(dataSet2, "Session");
        if ((int)dataSet2.Tables["Session"].Rows.Count > 0)
        {
            try
            {
                Session.SendToTarget(fixml, ssID);
                UpdateStatusToCompleted((int)
                    dataSet.Tables["Queue"].Rows[0]["id"]);
            }
            catch (SessionNotFound)
            {
                MoveToLastestQueue((int)
                    dataSet.Tables["Queue"].Rows[0]["id"]);
            }
        }
        else
        {
            MoveToLastestQueue((int)
                dataSet.Tables["Queue"].Rows[0]["id"]);
        }
}
```

บรรทัดที่ 3-12 เชื่อมต่อ MySQL และ Select Record ที่มีสถานะ Processed แล้ว (สถานะ Processes คือมีค่าข้อมูลใน Column outboundmsg และ Column status มีค่าเท่ากับ 0)

บรรทัดที่ 13-15 นำข้อมูลที่ Select มาเก็บในรูปแบบ DataSet

บรรทัดที่ 16-32 นำข้อมูลใน DataSet มาสร้าง Session และ FIX Message

บรรทัดที่ 33-43 เป็นการตรวจสอบว่า Client ที่จะทำการส่ง FIX Message ไปให้นั้นกำลังออนไลน์อยู่หรือเปล่า

บรรทัดที่ 44-62 หาก Client ออนไลน์อยู่จะเข้าไปทำงานตามคำสั่งในลูป if ตรงบรรทัดที่ 44-57 แต่ถ้าหากออฟไลน์ ก็จะเข้าไปทำงานในลูป else ตรงบรรทัดที่ 58-62

บรรทัดที่ 44-51 ส่ง FIX Message ไปที่ Client และเรียกใช้งาน Method UpdateStatusToCompleted เพื่อทำการ Update Column status ใน Record ที่ทำการ Select ออกมาเมื่อสักครู่อีกทีหนึ่งเป็นการบอกว่า Record นี้ได้ถูกส่งไปยังปลายทางเรียบร้อยแล้ว โดยรายละเอียดในMethod นี้จะอธายในข้อถัดไป

บรรทัดที่ 52-56 หากค่า Session ไม่ตรงกับที่กำหนดใน Config File เนื่องจากอาจถูกแก้ไข Config File ในระหว่างการทำงานของโปรแกรม Method MoveToLastestQueue ก็จะถูกเรียกใช้งาน

บรรทัดที่ 58-62 ถ้าหาก Client ออฟไลน์อยู่ ซึ่งหมายความว่าทาง Server ไม่สามารถส่ง FIX Message ไปยังปลายทางได้ ให้ทำการเรียกใช้ Method MoveToLastestQueue เพื่อย้าย Record นี้ไปที่ Row สุดท้ายของฐานข้อมูลเพื่อรอโปรแกรมมา Select อีกทีนรอบต่อไป โดยการทำงานใน Method MoveToLastestQueue จะอธิบายในข้อต่อไป

6. สร้าง Method UpdateStatusToCompleted และเขียนโค้ดการทำงานดังโค้ดด้านล่าง

```
1  public void UpdateStatusToCompleted(int input)
2  {
3      MySqlConnection connection = new MySqlConnection(connectionStr);
4      connection.Open();
5      string sqlStr = "UPDATE fix_message";
6          sqlStr += " SET status = " + 1;
7          sqlStr += " WHERE id = " + input;
8      MySqlCommand sqlCmd = new MySqlCommand(sqlStr, connection);
9      sqlCmd.ExecuteNonQuery();
10 }
```

กำหนดคำสั่ง SQL ให้ทำการ Update ใน Record ที่มี Column id เท่ากับ Parameter ที่ส่งเข้ามาใน Method นี้ และนำไป Query ที่ MySQL

7. สร้าง Method MoveToLastestQueue และเขียนโค้ดการทำงานดังโค้ดด้านล่าง

```
public void MoveToLastestQueue(int input)
{
    MySqlConnection connection = new MySqlConnection(connectionStr);
    connection.Open();
    string sqlStr = "SELECT * FROM fix_message";
    sqlStr += " WHERE id = " + input;
    MySqlCommand sqlCmd = new MySqlCommand(sqlStr, connection);
    MySqlDataAdapter adapter = new MySqlDataAdapter(sqlCmd);
    DataSet dataSet = new DataSet();
    adapter.Fill(dataSet, "Queue");
    beginString = dataSet.Tables["Queue"].Rows[0]["beginstring"].ToString();
    msgType = dataSet.Tables["Queue"].Rows[0]["msgtype"].ToString();
    senderComID = dataSet.Tables["Queue"].Rows[0]["sendercomid"].ToString();
    senderSubID = dataSet.Tables["Queue"].Rows[0][4].ToString();
    targetComID = dataSet.Tables["Queue"].Rows[0][5].ToString();
    targetSubID = dataSet.Tables["Queue"].Rows[0][6].ToString();
    xmlData = dataSet.Tables["Queue"].Rows[0][7].ToString();
    inboundMsg = dataSet.Tables["Queue"].Rows[0][8].ToString();
    outboundMsg = dataSet.Tables["Queue"].Rows[0][9].ToString();
    int status = (int)dataSet.Tables["Queue"].Rows[0][10];
    string sqlStr2 = "DELETE FROM fix_message WHERE id =" + input;
    MySqlCommand sqlCmd2 = new MySqlCommand(sqlStr2, connection);
    sqlCmd2.ExecuteNonQuery();
    string sqlStr3 = "INSERT INTO fix_message(beginstring, msgtype,
        sendercomid, sendersubid,";
        sqlStr3 += " targetcomid, targetsubid, xmlData, inboundmsg,
        outboundmsg, status)";
        sqlStr3 += " VALUES('" + beginString + "', '" + msgType + "', '" +
        senderComID + "',";
        sqlStr3 += " '" + senderSubID + "', '" + targetComID + "', '" +
        targetSubID + "',";
        sqlStr3 += " '" + xmlData + "', '" + inboundMsg + "', '" +
        outboundMsg + "', " + status + ")";
    MySqlCommand sqlCmd3 = new MySqlCommand(sqlStr3, connection);
    sqlCmd3.ExecuteNonQuery();
}
```

บรรทัดที่ 6-7 เชื่อมต่อ MySQL

บรรทัดที่ 5-7 กำหนดคำสั่ง SQL และส่งไป Query ที่ MySQL

บรรทัดที่ 8-20 ข้อมูลที่ได้จากการ Query ในบรรทัดที่ 7 มาเก็บใน DataSet

บรรทัดที่ 21-35 ย้าย Record ไป Row สุดท้ายด้วยวิธีการ Delete และ Insert ใหม่ด้วยข้อมูลใน DataSet

8. เรียกใช้งาน Method TaskScheduler ใน Method ServerApp

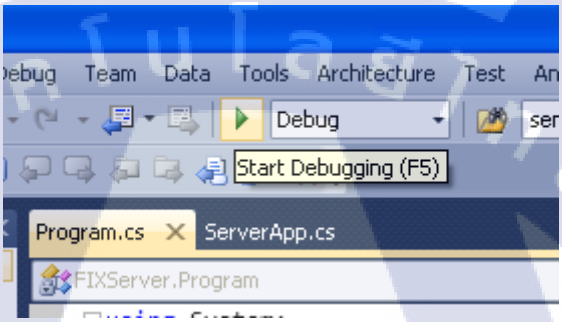
```
public ServerApp(SessionSettings settings)
{
    this.settings = settings;
    TaskScheduler();
}
```

บทที่ 4
ผลการดำเนินงาน การวิเคราะห์ และสรุปผลต่าง ๆ

4.1 ขั้นตอนและผลการดำเนินงาน

1. ทดสอบการสร้าง Session และการ Logon (MsgType A) ของ Client กับ Server

1. เปิดการทำงาน (Start Debugging) ของ Server ที่เขียนด้วย C# ในโปรแกรม Visual Studio ดังขั้นตอนในรูปที่ 4.1

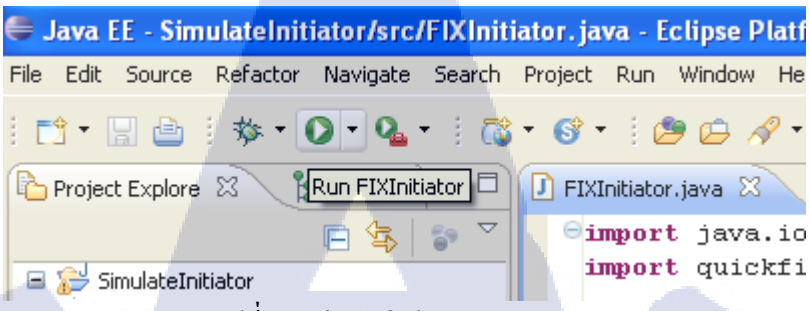


รูปที่ 4.1 วิธี Start Debugging ใน Microsoft Visual Studio 2010

2. หลังจากที่โปรแกรมสร้าง Session สำเร็จแล้ว Method OnCreate จะทำงานดังรูปที่ 4.2

รูปที่ 4.2 Console แสดงผลการทำงานของ Method OnCreate

3 . เปิดการทำงาน (Run) ของ Client ที่เขียนด้วย Java ในโปรแกรม Eclipse ดังขั้นตอนในรูปที่ 4.3



รูปที่ 4.3 วิธี Run ในโปรแกรม Eclipse

การทำงานของ Client นั้นจะเหมือนกับ Server คือ หลังจากที่สร้าง Session สำเร็จแล้ว โปรแกรมจะทำงานใน Method onCreate และพิมพ์ผลลัพธ์ออกมาใน Console ดังรูปที่ 4.4

```
On create : FIXT.1.1:024/1->SERVER1/01
To admin : FIXT.1.1:024/1->SERVER1/01
8=FIXT.1.1□9=89□35=A□34=1□49=024□50=1□52=20130923-02:53:14.356□56=SERVER1□57=01□98=0□
From admin : FIXT.1.1:024/1->SERVER1/01
8=FIXT.1.1□9=89□35=A□34=1□49=SERVER1□50=01□52=20130923-02:53:14.715□56=024□57=1□98=0□
On logon : FIXT.1.1:024/1->SERVER1/01
From admin : FIXT.1.1:024/1->SERVER1/01
8=FIXT.1.1□9=64□35=0□34=2□49=SERVER1□50=01□52=20130923-02:53:44.733□56=024□57=1□10=16
To admin : FIXT.1.1:024/1->SERVER1/01
8=FIXT.1.1□9=64□35=0□34=2□49=024□50=1□52=20130923-02:53:44.733□56=SERVER1□57=01□10=16
```

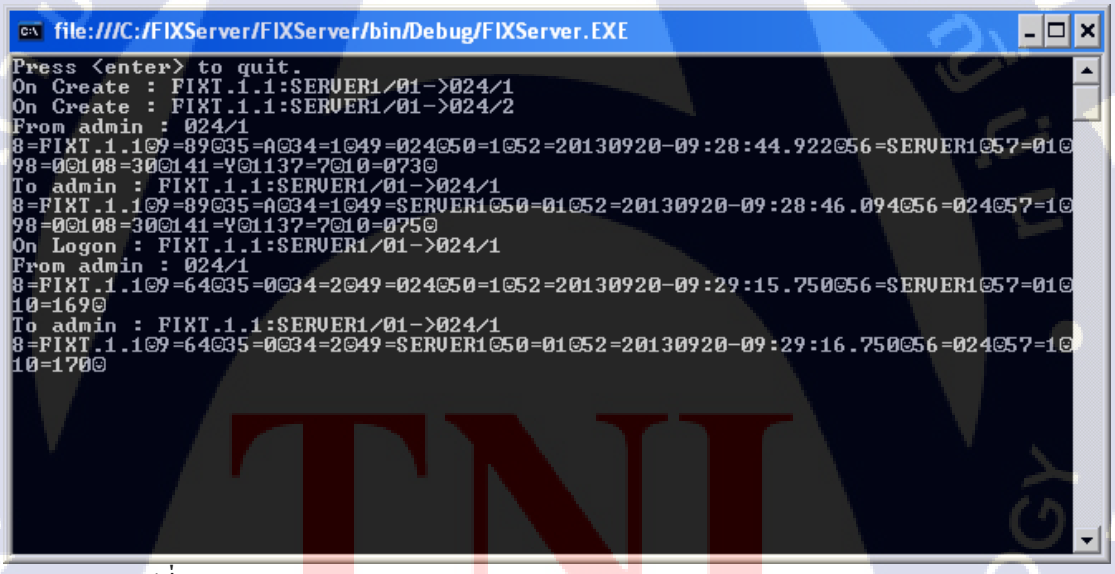
รูปที่ 4.4 Console แสดงผลการรับส่ง MsgType A และ 0 ของ Client ในโปรแกรม Eclipse

จากรูปที่ 4.4 หลังจากทำงานใน Method onCreate แล้ว Client จะส่ง MsgType A เพื่อขอทำการ Logon ไปที่ Sever ทันที หากทำการ Logon สำเร็จ โปรแกรมก็จะทำงานใน Method onLogon และมีการส่ง MsgType 0 (Heartbeat) กันระหว่าง Server กับ Client เพื่อเป็นการตรวจสอบว่ามีการเชื่อมต่อกันอยู่หรือไม่ตามระยะเวลาที่กำหนดใน Config File

ใน Method onCreate, onLogon และ toAdmin มีโค้ดดังนี้

```
@Override
public void onCreate(SessionID sessionID)
{
    System.out.println("On create : " + sessionID);
    this.sessionID = sessionID;
}
@Override
public void onLogon(SessionID sessionID)
{
    System.out.println("On logon : " + sessionID);
}
@Override
public void toAdmin(Message message, SessionID sessionID)
{
    System.out.println("To admin : " + sessionID + "\n" + message);
}
```

เมื่อสลับไปดูที่ Console ของ Server จะมีผลลัพธ์ ต่าง ๆ ถูกพิมพ์ออกมาตามโค้ดที่ได้เขียนไว้ใน Method OnLogon, ToAdmin และ FromAdmin ดังรูปที่ 4.5



รูปที่ 4.5 Console แสดงการรับ-ส่ง MsgType A (Logon) และ 0 (Heartbeat) ของ Server

โค้ดใน Method OnLogon, ToAdmin และ FromAdmin ของ Server มีดังนี้

```
1  public void OnLogon(SessionID sessionID)
2  {
3      Console.WriteLine("On Logon : " + sessionID.ToString());
4  }
5  public void ToAdmin(Message message, SessionID sessionID)
6  {
7      Console.WriteLine("To admin : " + sessionID + "\n" + message);
8  }
9  public void FromAdmin(Message message, SessionID sessionID)
10 {
11     msgType = message.Header.GetString(35);
12     if (msgType.Equals(MsgType.XML_NON_FIX))
13     {
14         beginString = message.Header.GetField(8);
15         senderComID = message.Header.GetField(49);
16         senderSubID = message.Header.GetField(50);
17         targetComID = message.Header.GetField(56);
18         targetSubID = message.Header.GetField(57);
19         xmlData = message.Header.GetField(213);
20         MySqlConnection connection = new MySqlConnection(connectionStr);
21         connection.Open();
22         string sqlStr = "INSERT INTO fix_message(beginstring, msgtype,";
23             sqlStr += " sendercomid, sendersubid, targetcomid, targetsubid,
24                 xmldata, inboundmsg)";
25             sqlStr += " VALUES('" + beginString + "', '" + msgType + "', '" +
26                 senderComID + "',";
27             sqlStr += " '" + senderSubID + "', '" + targetComID + "', '" +
28                 targetSubID + "',";
29             sqlStr += " '" + xmlData + "', '" + message.ToString().Replace(
30                 (char)1, '^') + "')";
31         MySqlCommand sqlCmd = new MySqlCommand(sqlStr, connection);
32         sqlCmd.ExecuteNonQuery();
33     }
34     Console.WriteLine("From admin : " + sessionID + "\n" + message);
35 }
```

จากโค้ดด้านบน หากส่ง MsgType A (Logon), 0 (Heartbeat) หรืออื่น ๆ ที่ไม่ใช่ n (XML non FIX) โปรแกรมจะพิมพ์ FIX Message ที่รับมาก Client ทาง Console เท่านั้น ไม่เข้าไปทำงานในลูป if ในบรรทัดที่ 12-33

2. ทดสอบการส่ง MsgType n (XML non FIX) ของโปรแกรม

1. ใน Client ได้เขียนโค้ดทดสอบการส่ง MsgType n ใน Method onLogon ดังโค้ดด้านล่างดังนี้

```java
@Override
public void onLogon(SessionID sessionID)
{
    System.out.println("On logon : " + sessionID);
    try
    {
        Message message = new Message();
        message.getHeader().setField(new quickfix.field.MsgType("n"));
        String xmlData = "<request><msgcd>1111</msgcd>" +
            "<rqid>2222</rqid></request>";
        message.setInt(212, xmlData.length());
        message.setString(213, xmlData);
        Session.sendToTarget(message, sessionID);
    }
    catch (SessionNotFound e)
    {
        System.out.print(e.toString());
    }
}
```

จากโค้ดด้านบน โปรแกรมจะส่ง FIX Message ไปให้ Server ทันทีหลังจากที่ทำการ Logon สำเร็จ โดยกำหนด MsgType (แท็ก 35) เป็น n และ XmlData (แท็ก 213) ดังนี้

<Request>

<MsgCd>1111></MsgCd>

<RqID>2222<RqID>

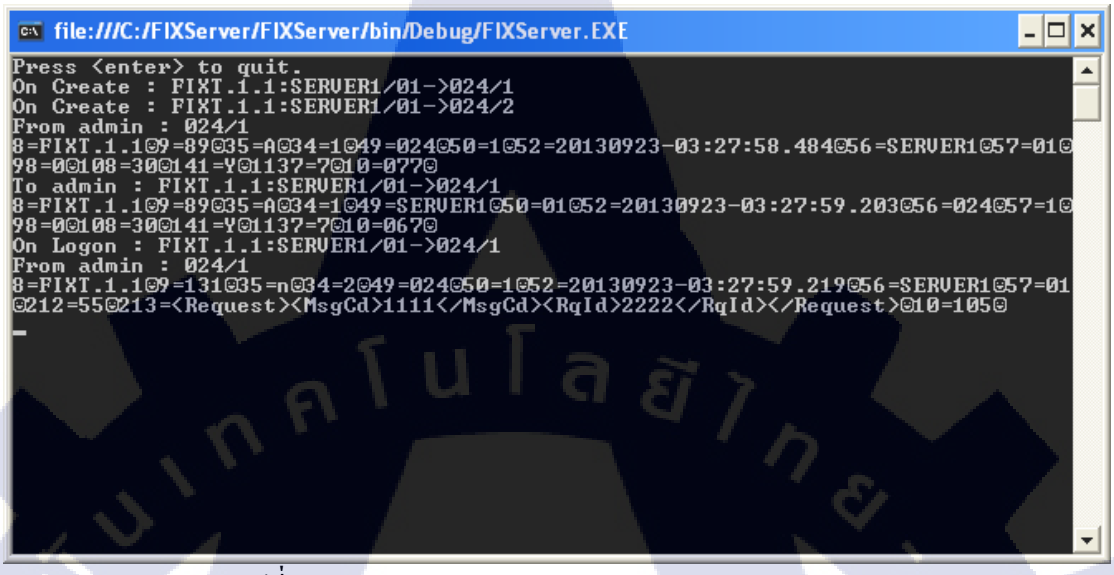
</Request>

และผลที่ได้ก็คือ Client ส่ง FIX Message ไปให้ Server สำเร็จและ Method toApp ถูกเรียกใช้งานดังรูปที่ 4.6

```
On logon : FIXT.1.1:024/1->SERVER1/01
To app : FIXT.1.1:024/1->SERVER1/01
8=FIXT.1.1□9=131□35=n□34=2□49=024□50=1□52=20130923-03:26:19.169□56=SERVER1□57=01□212
=55□213=<Request><MsgCd>1111</MsgCd><RqId>2222</RqId></Request>□10=105□
```

รูปที่ 4.6 Console แสดงผลการทดสอบส่ง MsgType n ไปให้ Server

เมื่อสลับไปดูที่ Console ของ Server โปรแกรมได้ทำการพิมพ์ FIX Message ที่รับมาจาก Client ผ่าน Method FromAdmin ดังรูปที่ 4.7 และทำการ Insert ลงในฐานข้อมูล ดังรูปที่ 4.8



รูปที่ 4.7 Consoleแสดงผลการทดสอบรับ MsgType n จาก Client

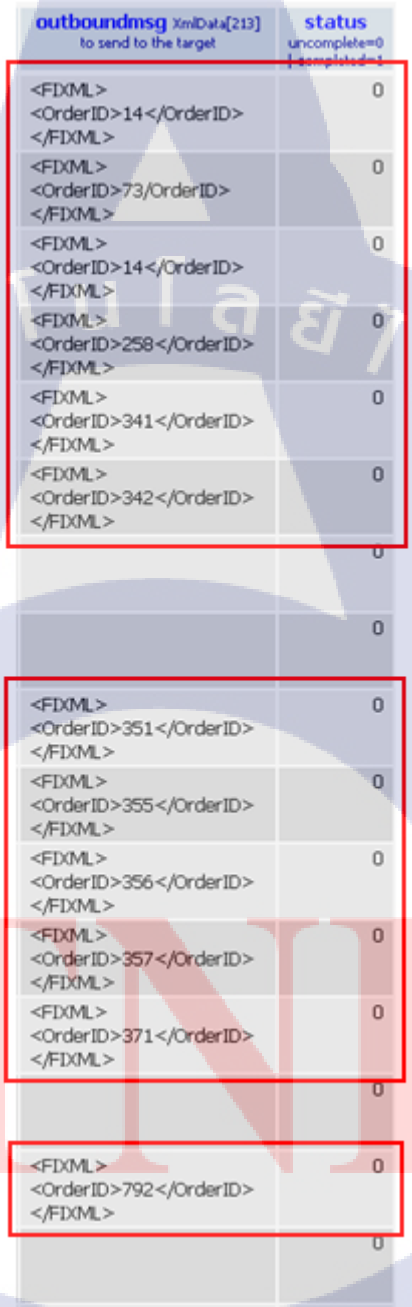
ฟิลด์	ชนิด	ค่า
id	int(50)	1693
beginstring	varchar(10)	FIXT.1.1
msgtype	varchar(2)	n
sendercomid	varchar(50)	024
sendersubid	varchar(50)	1
targetcomid	varchar(50)	SERVER1
targetsubid	varchar(50)	01
xmldata	text	<Request><MsgCd>1111</MsgCd> <RqId>2222</RqId></Request>
inboundmsg	text	8=FIXT.1.1^9=131^35=n^34=2^49=024 ^50=1^52=20130923- 03:27:59.219^56=SERVER1^57=01^212= 55^213=<Request> <MsgCd>1111</MsgCd> <RqId>2222</RqId> </Request>^10=105^
outboundmsg	text	
status	int(1)	0
		ลงมือ

รูปที่ 4.8 ผลการทดสอบนำ MsgType n ที่รับมาแล้ว Insert ลงใน MySQL

3. ทดสอบส่ง FIX Message จาก Server ไปให้ Client

วิธีการคือ Server จะหยิบ Record ที่มีสถานะ Prosessed (มีข้อมูลใน Column outboundmsg และ Column status เท่ากับ 0) ส่งไปให้ Client ทุก ๆ 10 วินาที ดังโค้ดที่ได้อธิบายไปในบทที่ 3

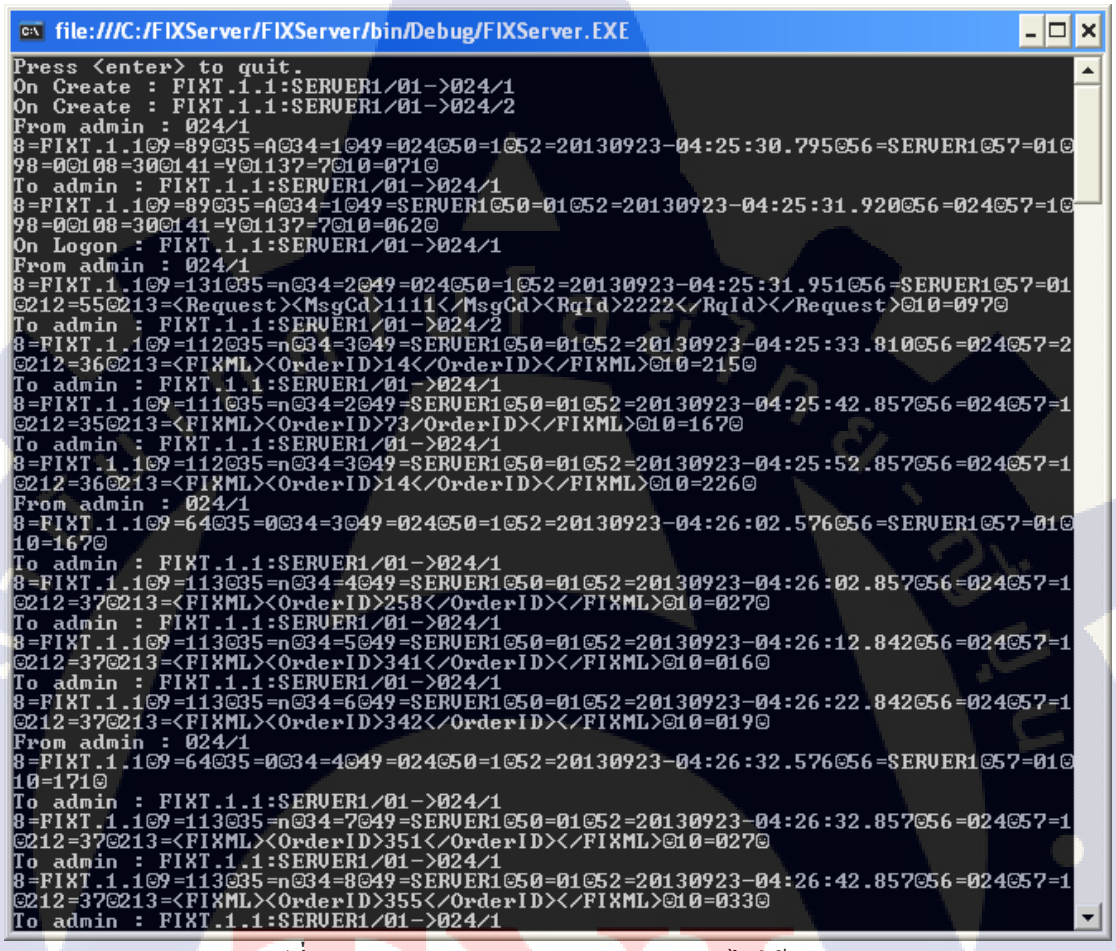
ก่อนเริ่มทำการทดสอบ ได้เตรียมฐานข้อมูลตามเงื่อนไขที่โปรแกรมจะหยิบ Record ไปปั้นเป็น FIX Message และส่งไปให้ Client ซึ่งเงื่อนไขก็คือ มีข้อมูลใน Column outboundmsg และ Column status เท่ากับ 0 ดังรูปที่ 4.9



รูปที่ 4.9 ตัวอย่าง Record ที่ Server หยิบไปปั้นเป็น FIX Message

จากรูปที่ 4.9 Record ที่ตรงกับเงื่อนไขก็คือส่วนที่อยู่ในกรอบสีแดง

หลังจากทำการทดสอบโปรแกรม หน้าจอ Console ของ Server แสดงผลออกมาดังรูปที่ 4.10 ส่วนหน้าจอ Console ของ Client แสดงผลออกมาดังรูปที่ 4.11 ผลลัพธ์ในฐานนข้อมูลปรากฏดังรูปที่ 4.12 ดังนี้



รูปที่ 4.10 Console แสดงผลการส่ง MsgType n ไปให้ Client

```
On create : FIXT.1.1:024/1->SERVER1/01
To admin : FIXT.1.1:024/1->SERVER1/01
8=FIXT.1.1□9=89□35=A□34=1□49=024□50=1□52=20130923-04:25:30.795□56=SERVER1□57=01□98=0
From admin : FIXT.1.1:024/1->SERVER1/01
8=FIXT.1.1□9=89□35=A□34=1□49=SERVER1□50=01□52=20130923-04:25:31.920□56=024□57=1□98=0
On logon : FIXT.1.1:024/1->SERVER1/01
To app : FIXT.1.1:024/1->SERVER1/01
8=FIXT.1.1□9=131□35=n□34=2□49=024□50=1□52=20130923-04:25:31.951□56=SERVER1□57=01□212
From app : FIXT.1.1:024/1->SERVER1/01
8=FIXT.1.1□9=111□35=n□34=2□49=SERVER1□50=01□52=20130923-04:25:42.857□56=024□57=1□212
From app : FIXT.1.1:024/1->SERVER1/01
8=FIXT.1.1□9=112□35=n□34=3□49=SERVER1□50=01□52=20130923-04:25:52.857□56=024□57=1□212
To admin : FIXT.1.1:024/1->SERVER1/01
8=FIXT.1.1□9=64□35=0□34=3□49=024□50=1□52=20130923-04:26:02.576□56=SERVER1□57=01□10=1
From app : FIXT.1.1:024/1->SERVER1/01
8=FIXT.1.1□9=113□35=n□34=4□49=SERVER1□50=01□52=20130923-04:26:02.857□56=024□57=1□212
From app : FIXT.1.1:024/1->SERVER1/01
8=FIXT.1.1□9=113□35=n□34=5□49=SERVER1□50=01□52=20130923-04:26:12.842□56=024□57=1□212
From app : FIXT.1.1:024/1->SERVER1/01
8=FIXT.1.1□9=113□35=n□34=6□49=SERVER1□50=01□52=20130923-04:26:22.842□56=024□57=1□212
To admin : FIXT.1.1:024/1->SERVER1/01
8=FIXT.1.1□9=64□35=0□34=4□49=024□50=1□52=20130923-04:26:32.576□56=SERVER1□57=01□10=1
From app : FIXT.1.1:024/1->SERVER1/01
8=FIXT.1.1□9=113□35=n□34=7□49=SERVER1□50=01□52=20130923-04:26:32.857□56=024□57=1□212
From app : FIXT.1.1:024/1->SERVER1/01
8=FIXT.1.1□9=113□35=n□34=8□49=SERVER1□50=01□52=20130923-04:26:42.857□56=024□57=1□212
From app : FIXT.1.1:024/1->SERVER1/01
8=FIXT.1.1□9=113□35=n□34=9□49=SERVER1□50=01□52=20130923-04:26:52.842□56=024□57=1□212
```

รูปที่ 4.11 Console แสดงผลการรับ MsgType n ของ Client

outboundmsg XmlData[213] to send to the target	status uncomplete=0 | completed=1
<FIXML><OrderID>14</OrderID></FIXML>	0
<FIXML><OrderID>73/OrderID></FIXML>	0
<FIXML><OrderID>14</OrderID></FIXML>	0
<FIXML><OrderID>258</OrderID></FIXML>	0
<FIXML><OrderID>341</OrderID></FIXML>	0
<FIXML><OrderID>342</OrderID></FIXML>	0
	0
	0
<FIXML><OrderID>351</OrderID></FIXML>	0
<FIXML><OrderID>355</OrderID></FIXML>	0
<FIXML><OrderID>356</OrderID></FIXML>	0
<FIXML><OrderID>357</OrderID></FIXML>	0
<FIXML><OrderID>371</OrderID></FIXML>	0
	0
<FIXML><OrderID>792</OrderID></FIXML>	0
	0

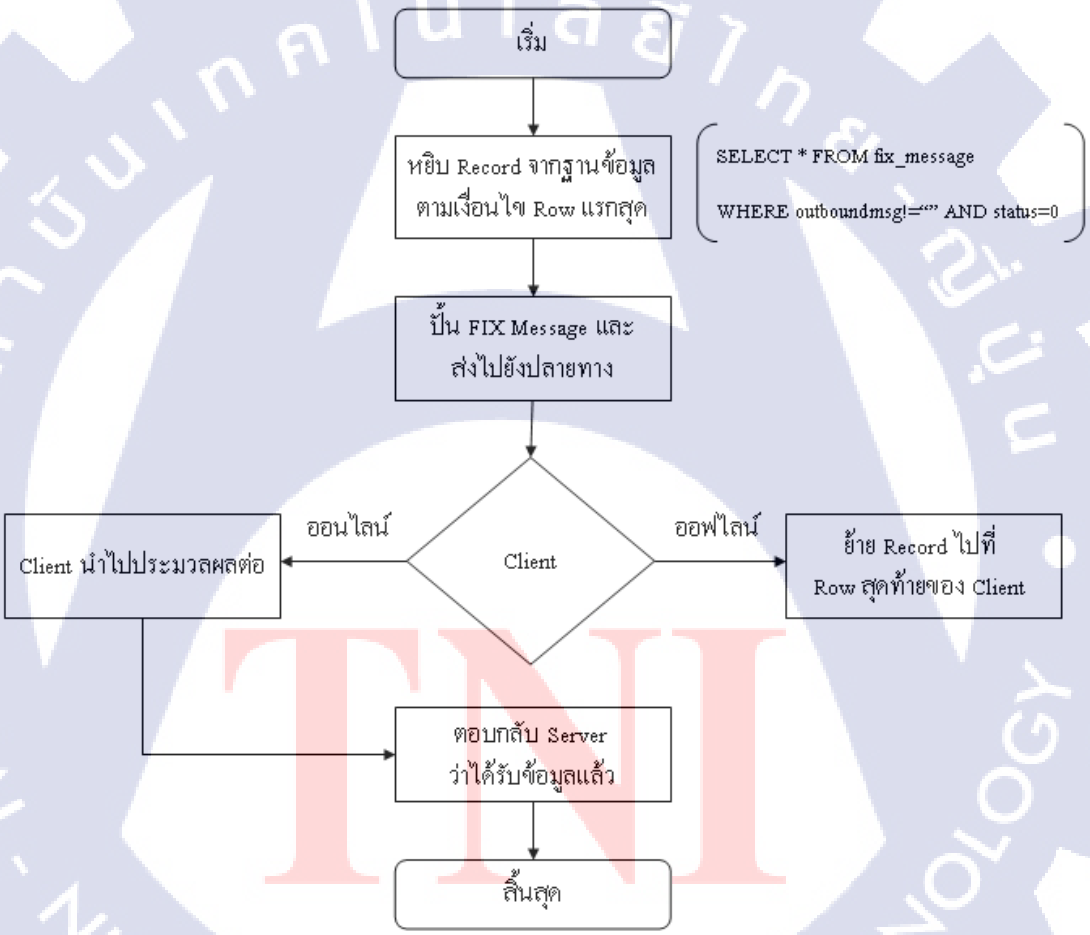
outboundmsg XmlData[213] to send to the target	status uncomplete=0 | completed=1
<FIXML><OrderID>14</OrderID></FIXML>	1
<FIXML><OrderID>73/OrderID></FIXML>	1
<FIXML><OrderID>14</OrderID></FIXML>	1
<FIXML><OrderID>258</OrderID></FIXML>	1
<FIXML><OrderID>341</OrderID></FIXML>	1
<FIXML><OrderID>342</OrderID></FIXML>	1
	0
	0
<FIXML><OrderID>351</OrderID></FIXML>	1
<FIXML><OrderID>355</OrderID></FIXML>	1
<FIXML><OrderID>356</OrderID></FIXML>	1
<FIXML><OrderID>357</OrderID></FIXML>	1
<FIXML><OrderID>371</OrderID></FIXML>	1
	0
<FIXML><OrderID>792</OrderID></FIXML>	1
	0

รูปที่ 4.12 แสดง Record ใน Table fix_message ก่อนทำการทดสอบ (ซ้ายมือ) และหลังทำการทดสอบ (ขวามือ) ส่ง FIX Message

จากรูปที่ 4.12 หลังจากที่ Server ส่ง MsgType n ไปให้ Client แล้ว Column status จะถูก Update เป็น 1 ซึ่งหมายความว่า Record นั้น ๆ ถูก Server นำข้อมูลใน Column outboundmsg ไปปั้นเป็น FIX Message และส่งไปยังปลายทางเรียบร้อยแล้วโดยไม่มี Error ใด ๆ

4. ทดสอบส่ง FIX Message ในขณะที่ Client ออฟไลน์อยู่

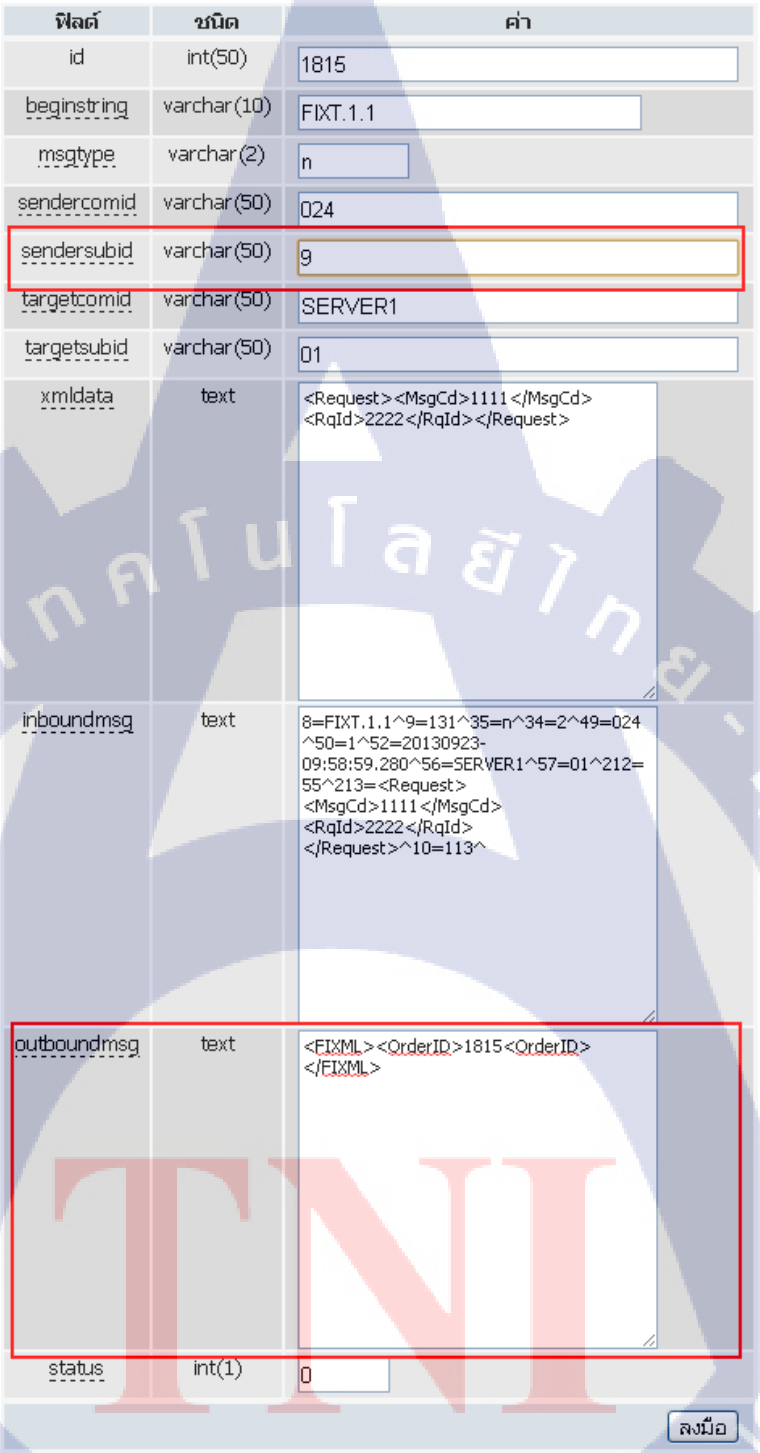
วัตถุประสงค์ในการทดสอบหัวข้อนี้ก็คือ เมื่อ Server หยิบ Record ที่มีสถานะ Proesssed จากฐานข้อมูลมาทำการปั้น FIX Message และส่งไปยังปลายทางต่าง ๆ หากปลายทางหรือ Client นั้น ๆ ไม่สามารถรับ FIX Message ที่ส่งมาได้ เนื่องจากออฟไลน์อยู่ไม่ว่าด้วยสาเหตุอะไรก็ตาม ก็ให้โปรแกรมตี FIX Message ที่ส่งไปนั้นกลับมา แล้วย้าย Record นั้นไปต่อ Row สุดท้ายใน Table (ไม่ต้อง Update Column status) เพื่อรอ Server มาหยิบหรือเรียกใช้ต่อไป และการที่ย้ายไปไว้ Record สุดท้ายนั้นก็เพื่อให้โปรแกรมสามารถหยิบข้อมูลใน Record ถัด ๆ ไปมาประมวลผลต่อได้โดยไม่สะดุด



รูปที่ 4.13 แผนภาพแสดงการทำงานใน Method SendMessage

ส่วนวิธีทดสอบมีดังนี้

1. ใน Table fix_message ให้แก้ไข Column sendercomid หรือ sendersubid ก็ได้ โดยแก้เป็นค่าที่ไม่ได้ระบุไว้ใน Config File



รูปที่ 4.14 Record ที่ใช้ไปทดสอบเกี่ยวกับปัญหาเรื่อง Client ออฟไฟลน์ขณะที่ส่ง FIX Message ไปให้

จากรูปที่ 4.14 ได้แก้ไข Column sendersubid เป็น 9 ซึ่งไม่ถูกระบุใน Config File ของ Server และระบ XmlData ที่จะส่งไปให้ปลายทางเป็น

<FIXML><OrderID>1815</OrderID></FIXML> และทำการทดสอบผลการทำงาน โดยผลที่ได้ปรากฏดังรูปที่ 4.15

ฟิลด์	ชนิด	ค่า
id	int(50)	1896
beginstring	varchar(10)	FIXT.1.1
msgtype	varchar(2)	n
sendercomid	varchar(50)	024
sendersubid	varchar(50)	9
targetcomid	varchar(50)	SERVER1
targetsubid	varchar(50)	01
xmldata	text	<Request><MsgCd>1111</MsgCd> <RqId>2222</RqId></Request>
inboundmsg	text	8=FIXT.1.1^9=131^35=n^34=2^49=024^50=1^52=20130923-09:58:59.280^56=SERVER1^57=01^212=55^213=<Request> <MsgCd>1111</MsgCd> <RqId>2222</RqId> </Request>^10=113^
outboundmsg	text	<FIXML><OrderID>1815<OrderID> </FIXML>
status	int(1)	0

ลงมือ

รูปที่ 4.15 ผลการทดสอบย้าย Record ไปที่ Row สุดท้ายของ Table fix_message

จากรูปที่ 4.15 ผลที่ได้จากการทดสบ คือ Record นี้ถูก Delete แล้วนำไป Insert ใน Row สุดท้ายของ Table โดยสั่งเกตุได้จากค่าไอดีที่เปลี่ยนจาก 1815 เป็น 1896 ซึ่งค่า id มีคุณสมบัติเป็น AUTO_INCREMENT และ 1896 เป็น Row สุดท้ายใน Table ขณะที่ทดสอบผลครั้งนี้

5. Data Dictionary ของฐานข้อมูล

ฐานข้อมูลที่ใช้ทดสอบนี้มีชื่อว่า quickfix และในฐานข้อมูลมี 2 Table โดยมีรายละเอียดดังนี้

Table fix_message

ใช้เก็บ FIX Message ที่รับมาจาก Client มี 11 Column ดังนี้

No.	Column Name	Data Type	Description	Example
1	id	int(50)	มีคุณบัติเป็น Primary Key และ AUTO_INCREMENT	1,2,3,…,n
2	beginstring	varchar(10)	FIX Version ที่ใช้สื่อสารกันระหว่าง Client กับ Server	FIXT.1.1
3	msgtype	varchar(2)	MsgType ของ FIX Message	n
4	sendercomid	varchar(50)	รหัสเครื่อง Client ที่รับ-ส่ง Message	Server
5	sendersubid	varchar(50)	รหัสเครื่อง Sub Client ที่รับ-ส่ง Message	001
6	targetcomid	varchar(50)	รหัสเครื่อง Server ที่รับ-ส่ง Message	Client
7	targetsubid	varchar(50)	รหัสเครื่อง Sub Server ที่รับ-ส่ง Message	<FIXML> <RqID>1<RqID> </FIXML>
8	xmldata	text	XML Data (Tag 213) ที่แกะมาจาก FIX Message	XML Data
9	inbounmsg	text	FIX Message ที่รับมา	8=FIXT.1.1^ 9=66^35=n^… ^10=012^
10	outboundmsg	text	XML Data ที่จะนำไปปั้นเป็น	<FIXML>

			FIX Message แล้วส่งให้ Client	<RqID>1<RqID> </FIXML>
11	stattus	int(50)	สถานะของข้อมูล มี 2 ค่า คือ 0 หมายถึง New Order และ 1 หมายถึง Processed	0, 1

ตารางที่ 4.1 Data Dictionary ของ Table fix_message

Table fix_message

ใช้เก็บค่า SessionID ที่ Client ที่กำลังเชื่อมต่อกับ Server อยู่ มี 7 Column ดังนี้

No.	Column Name	Data Type	Description	Example
1	id	int(50)	มีคุณบัติเป็น Primary Key และ AUTO_INCREMENT	1,2,3,…,n
2	beginstring	varchar(10)	FIX Version ที่ใช้สื่อสารกันระหว่าง Client กับ Server	FIXT.1.1
3	msgtype	varchar(2)	MsgType ของ FIX Message	n
4	sendercomid	varchar(50)	รหัสเครื่อง Client ที่รับ-ส่ง Message	Server
5	sendersubid	varchar(50)	รหัสเครื่อง Sub Client ที่รับ-ส่ง Message	001
6	targetcomid	varchar(50)	รหัสเครื่อง Server ที่รับ-ส่ง Message	Client
7	targetsubid	varchar(50)	รหัสเครื่อง Sub Server ที่รับ-ส่ง Message	<FIXML> <RqID>1<RqID> </FIXML>

ตารางที่ 4.2 Data Dictionary ของ Table session

4.2 ผลการวิเคราะห์ข้อมูล

ผลการทดสอบเขียนโปรแกรมจัดการ FIX Message ร่วมกับฐานข้อมูล MySQL โดยกำหนดให้ ฝั่ง Server เป็น .NET Platform และฝั่ง Client เป็น Java Platform สามารถทำงานร่วมกันได้ และสามารถรองการรับ-ส่ง FIX Message จำนวนมาก ๆ ได้

4.4 วิเคราะห์และวิจารณ์ข้อมูล โดยเปรียบเทียบผลที่ได้รับกับวัตถุประสงค์และจุดมุ่งหมายในการจัดทำโครงงาน

ที่	วัตถุประสงค์	ผลที่ได้รับจากการดำเนินการ
1	เขียนโปรแกรม FIX Application ให้เชื่อมต่อกันระหว่าง Server ที่เป็น .NET และ Client ที่เป็น Java	สามารถเชื่อมต่อกันได้ตามวัตถุประสงค์
2	เขียนโปรแกรมทดสอบรับค่า SessionID ที่ Logon มายัง Server แล้วนำมา Insert ลงฐานข้อมูล และ Delete เมื่อ Client Logout เพื่อเป็นการบอกว่ามี Client ใดบ้างที่เชื่อมต่อกับ Server อยู่ในขณะนั้น	สามารถจัดการข้อมูลได้ตามวัตถุประสงค์
3	เขียนโปรแกรมทดสอบการส่ง FIX Message จาก Client ไปให้ Server	สามารถส่ง FIX Message ไปให้ Server ได้ตามวัตุประสงค์
4	เขียนโปรแกรมทดสอบการส่ง FIX Message ไปให้ Client โดยให้ Server ดึง Record ที่ได้กำหนดเงื่อนไขไว้จากฐานข้อมูลมาสร้างเป็น FIX Message แล้วส่งไป	สามารถสร้าง FIX Message ตามที่ระบุไว้ใน Record และส่งไปยังปลายทางได้ตามวัตถุประสงค์
5	เขียนโปรแกรมทดสอบการส่ง FIX Message โดยดึง Record จากฐานข้อมูลมาประมวลผลด้วยหลักการคิว (Queue) แบบ First In First Out	โปรแกรมสามารถทำงานได้ตามรูปแบบที่ออกแบบไว้
6	เขียนโปรแกรมแก้ไขปัญหาเกี่ยวกับการจัดการ FIX Message นั้น ๆ หาก Client ออฟไลน์ขณะที่ส่ง FIX Message ไปให้	หากเจอปัญหาดังกล่าว ได้เขียนโปรแกรมให้ย้าย Record นั้นไปไว้ Row สุดท้ายของ Table เพื่อรอ Server หยิบไปประมวลผลต่อไป
7	เขียนโปรแกรมแก้ไขปัญหาเกี่ยวกับการจัดการ FIX Message หากส่ง FIX Message ไปแล้วเกิด Session Not Found	หากเจอปัญหาดังกล่าว ได้เขียนโปรแกรมให้ย้าย Record นั้นไปไว้ Row สุดท้ายของ Table เพื่อรอ Server หยิบไปประมวลผลต่อไป

8	เขียนโปรแกมทดสอบการรับ FIX Message จาก Client มา Insert ลงฐานข้อมูล MySQL	สามารถนำ FIX Message ที่รับจาก Client มา Insert ลงฐานข้อมูลได้ตามวัตถุประสงค์

ตารางที่ 4.3 วิเคราะห์และวิจารณ์ข้อมูล โดยเปรียบเทียบผลที่ได้รับกับวัตถุประสงค์และจุดมุ่งหมายในการจัดทำโครงงาน

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 สรุปผลการดำเนินโครงงาน

ผลจากการพัฒนาโปรแกรมจัดการ FIX Message ด้วยคิว และฐานข้อมูล MySQL สรุปผลดังรายละเอียดต่อไปนี้

1. Server ที่เป็น .NET Platform และ Client ที่ปืน Java Platform สามารถรับ-ส่ง FIX Message หากันได้

2. ในส่วนของการจัดการ Session ของ Client ที่เชื่อมต่อมาที่ Server และการจัดการรับ-ส่ง FIX Message นั้นสามารถใช้ MySQL มาเป็นตัวกลางในการจัดการข้อมูลดังกล่าวได้

3. โปรแกรมสามารถจัดการการรับ-ส่ง FIX Message ด้วย Queue แบบ FIFO ร่วมกับฐานข้อมูล MySQL ได้โดยไม่มีปัญหาใด ๆ

5.2 ปัญหาแนวทางการแก้ไขปัญหา

ที่	ปัญหา	แนวทางแก้ไขปัญหา
1.	เกิด Error เมื่อทดสอบการทำงานของโปรแกรม หรือโปรแกรมทำงานผิดพลาด	ตรวจสอบและแก้ไขข้อผิดพลาดโดยใช้ Debug Mode ของแต่ละซอฟต์แวร์ จะช่วยให้เราหาจุดที่มีข้อผิดพลาดได้รวดเร็วขึ้น หรือการใช้คำสั่ง try…catch…finally
2.	การทำความเข้าใจเกี่ยวกับระบบงานที่ใช้อยู่ในปัจจุบัน เนื่องจากมีความซับซ้อนและมีบางจุดที่เข้าใจได้ยาก	ขอคำแนะนำจากพนักงานที่ปรึกษาเพื่อให้ช่วยอธิบายส่วนที่ไม่เข้าใจ
3.	การเขียนโปรแกรมหรือใช้โปรแกรมที่ไม่เคยใช้หรือศึกษามาก่อนหน้านี้ เช่น ภาษา C#, โปรแกรม Microsoft Visual Studio	ค้นคว้าหาความรู้ในหนังสือ บทความในอินเตอร์เน็ต และวีดีโอการสอนต่าง ๆ ใน Youtube
4.	การเขียนโปรแกรม FIX Aplication เนื่องจากเอกสารที่ใช้ประกอบการศึกษาเป็นภาษาอังกฤษ	โหลดโปรแกรมตัวอย่างจากอินเตอร์เน็ตมาลองศึกษาประกอบ หรือขอคำแนะนำจากพนักงานที่ปรึกษาให้ช่วยอธิบายในส่วนที่ไม่

		เข้าใจ
5.	การรับ-ส่ง FIX Message ที่เป็นภาษาไทย เนื่องจากโปรแกรมไม่รองรับภาษาไทย	ต้องทำการ Encoding เป็น Base64 ก่อนส่งไปยังปลายทาง และให้ปลายทางทำการ Decoding ออกมา

ตารางที่ 5.1 ปัญหาละแนวทางแก้ไข

5.3 ข้อเสนอแนะจากการดำเนินงาน

1. ในการปฏิบัติงาน ต้องมีทักษะพื้นฐานเกี่ยวกับภาษา Programming แบบ OOP โดยเฉพาะภาษา C# และ Java และความรู้เกี่ยวกับซอฟต์แวร์ฐานข้อมูลต่าง ๆ อย่างเช่น Microsoft SQL Server, Oracle และ MySQL

2. เอกสารที่ใช้ในการศึกษาประกอบการเขียนโปรแกรม ส่วนใหญ่จะเป็นภาษาอังกฤษ จึงควรมีทักษะเกี่ยวกับการอ่านพอสมควร

3. การเขียนโปรแกรมควรตั้งชื่อตัวแปรให้ให้สื่อความหมาย และเขียนคอมเมนต์อธิบายการทำงานในส่วนต่างของโค้ด เพื่อให้ผู้อื่นที่จะนำไปพัฒนาต่อสามารถเข้าใจได้รวดเร็วขึ้น

เอกสารอ้างอิง

[1] ตลาดหลักทรัพย์แห่งประเทศไทย, **ภาพรวม ตลท.** [Online], Available: http://www.set.or.th/th/about/overview/history_p1.html [2556, กันยายน 26]

[2] Brian Driscoll, **Intro to the the FIX Protocol** [Online], Availiable: http://www.slideshare.net/BrianDriscoll/barcampnyc3-intro-to-fix [2556, กันยายน 26]

[3] QuickFIX/n**, QuickFIX/n** [Online], Available: http://quickfixn.org [2556, กันยายน 26]

[4] QuickFIX/J**, QuickFIX/J** [Online], Available: http://quickfixj.org [2556, กันยายน 26]

[5] Prof. Dr. Bernd Ulmann, **FIX Protpcol Basics** [Online], Available: http://www.fafner.dyndns.org/~vaxman/publications/fix_basics.pdf [2556, กันยายน 26]

[6] Onix Solutions Limited**, FIX Dictionary** [Online], Availiable: http://www.onixs.biz/fix-dictionary/ [2556, กันยายน 26]

[7] FIX Procotol Ltd, **FIX Interactive Message and Tag Explorer** [Online], Available: http://www.fixtradingcommunity.org/FIXimate/FIXimate3.0/ [2556, กันยายน 26]

[8] จักรกฤษณ์ แร่ทอง, **รู้จัก XML เบื้องต้น** [Online], Availiable: http://www.moe.go.th/moe/upload/itnews/htmlfiles/8952-8955.html [2556, กันยายน 26]

[9] Waschalapong Sopap, **ภาษา C# คืออะไร** [Online], Availiable: http://www.blogger.com/profile/16795505757249413376

[12] Wikipedia, **ไมโครซอฟท์ วิชวลสตูดิโอ** [Online], Availiable: http://th.wikipedia.org/wiki/ไมโครซอฟต์_วิชวลสตูดิโอ

[13] บัญชา ปะสีละเตสัง, 2554, **พัฒนาแอปพลิเคชั่นด้วย Visual C#**, พิมพ์ครั้งที่ 1, สำนักพิมพ์ซีเอ็ด.

[14] รศ. ธีรวัฒน์ ประกอบผล, 2555, **คู่มือการเขียนโปรแกรมเชิงวัตถุ JAVA OOP ฉบับสมบูรณ์**, พิมพ์ครั้งที่ 1, บริษัท วี.พริ้นท์ (1991) จำกัด.

ภาคผนวก

ตัวอย่างโค้ดบางส่วนใน FIX Application ที่ได้ทำการเขียนโปรแกรม

FIX Application ของ Server ที่เขียนด้วย C# ในโปรแกรม Visual Studio 2010

ไฟล์ Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using QuickFix;

namespace FIXServer
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Press <enter> to quit.");
            string configFile = @"C:\FIXServer\FIXServer\config\Server.cfg";
            SessionSettings sessionSettings = new SessionSettings(configFile);
            IApplication application = new ServerApp(sessionSettings);
            IMessageStoreFactory storeFactory = new
FileStoreFactory(sessionSettings);
            ILogFactory screenFactory = new ScreenLogFactory(false, false,
false);
            ILogFactory logFactory = new FileLogFactory(sessionSettings);
            IAcceptor server = new ThreadedSocketAcceptor(application,
                storeFactory, sessionSettings, logFactory);
            server.Start();
            Console.ReadKey();
            server.Stop();
        }
    }
}
```

ไฟล์ ServerApp.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using QuickFix;
using QuickFix.Fields;
using MySql.Data.MySqlClient;
using System.Timers;
using System.Data;

namespace FIXServer
{
    class ServerApp : MessageCracker, IApplication
    {
        private SessionSettings settings;
        private SessionID sessionID;
        // FIX fields except for inboundmsg and outboundmsg
        string msgType, beginString, senderComID, senderSubID, targetComID,
targetSubID, xmlData, inboundMsg, outboundMsg;
        string connectionStr = "SERVER=localhost;" +
             "DATABASE=quickfix;" +
             "UID=wichunun;" +
             "PASSWORD=123456;";

        public ServerApp(SessionSettings settings)
```

```
        {
            this.settings = settings;
            TaskScheduler();
        }

        public string Base64ToString(string input)
        {
            byte[] xmlDataBase64 = Convert.FromBase64String(input);
            string xmlDataUTF8 = Encoding.UTF8.GetString(xmlDataBase64);
            return xmlDataUTF8;
        }

        public string StringToBase64(string input)
        {
            byte[] xmlDataUTF8 = System.Text.Encoding.UTF8.GetBytes(input);
            string xmlDataStr = Convert.ToBase64String(xmlDataUTF8);
            return xmlDataStr;
        }

        public void TaskScheduler()
        {
            Timer timer = new Timer { Interval = 10000, Enabled = true };
            timer.Elapsed += new ElapsedEventHandler(SendMessage);
        }

        public void SendMessage(object sender, ElapsedEventArgs e)
        {
            MySqlConnection connection = new MySqlConnection(connectionStr);
            connection.Open();
            // SELECT reccord status = 0
            string sqlStr = "SELECT id, beginstring, msgtype, sendercomid,
sendersubid,";
            sqlStr += " targetcomid, targetsubid, outboundmsg";
            sqlStr += " FROM fix_message";
            sqlStr += " WHERE outboundmsg != '' AND status =" + 0;
            sqlStr += " ORDER BY id";
            sqlStr += " LIMIT 0, 1";
            MySqlCommand sqlCmd = new MySqlCommand(sqlStr, connection);
            MySqlDataAdapter adapter = new MySqlDataAdapter(sqlCmd);
            DataSet dataSet = new DataSet();
            adapter.Fill(dataSet, "Queue");

            // id[0] | beginstring[1] | msgtype[2] | sendercomid[3] |
sendersubid[4] |
            // targetcomid[5] | targetsubid[6] | outboundmsg[7]
            if ((int)dataSet.Tables["Queue"].Rows.Count > 0)
            {
                // Create FIX Message
                beginString =
dataSet.Tables["Queue"].Rows[0]["beginstring"].ToString();
                senderComID =
dataSet.Tables["Queue"].Rows[0]["targetcomid"].ToString();
                senderSubID = dataSet.Tables["Queue"].Rows[0][6].ToString();
                targetComID = dataSet.Tables["Queue"].Rows[0][3].ToString();
                targetSubID = dataSet.Tables["Queue"].Rows[0][4].ToString();
                outboundMsg = dataSet.Tables["Queue"].Rows[0][7].ToString();

                QuickFix.Message fixml = new QuickFix.Message();
                MsgType msgType = new MsgType("n");
                fixml.Header.SetField(msgType);
                xmlData = outboundMsg;
                fixml.Header.SetField(new XmlDataLen(xmlData.Length)); // 212
```

```
                fixml.Header.SetField(new XmlData(xmlData)); // 213
                // Create SessionID
                SessionID ssID = new SessionID(beginString, senderComID,
senderSubID, targetComID, targetSubID);

                // Check Session
                string sqlStr2 = "SELECT * FROM session";
                sqlStr2 += " WHERE beginstring = '" + beginString + "' AND";
                sqlStr2 += " sendercomid = '" + senderComID + "' AND";
                sqlStr2 += " sendersubid = '" + senderSubID + "' AND";
                sqlStr2 += " targetcomid = '" + targetComID + "' AND";
                sqlStr2 += " targetsubid = '" + targetSubID + "'";
                MySqlCommand sqlCmd2 = new MySqlCommand(sqlStr2, connection);
                // Use SqlDataAdapter & Dataset
                MySqlDataAdapter adapter2 = new MySqlDataAdapter(sqlCmd2);
                DataSet dataSet2 = new DataSet();
                adapter2.Fill(dataSet2, "Session");

                if ((int)dataSet2.Tables["Session"].Rows.Count > 0)
                {
                    try
                    {
                        Session.SendToTarget(fixml, ssID);
                        // Update status from 0 (uncomplete) to 1 (completed)

UpdateStatusToCompleted((int)dataSet.Tables["Queue"].Rows[0]["id"]);
                    }
                    catch (SessionNotFound)
                    {

MoveToLastestQueue((int)dataSet.Tables["Queue"].Rows[0]["id"]);
                    }
                }
                else
                {

MoveToLastestQueue((int)dataSet.Tables["Queue"].Rows[0]["id"]);
                }
        }

        public void UpdateStatusToCompleted(int input)
        {
            MySqlConnection connection = new MySqlConnection(connectionStr);
            connection.Open();
            string sqlStr = "UPDATE fix_message";
            sqlStr += " SET status = " + 1;
            sqlStr += " WHERE id = " + input;
            MySqlCommand sqlCmd = new MySqlCommand(sqlStr, connection);
            sqlCmd.ExecuteNonQuery();
        }

        public void MoveToLastestQueue(int input)
        {
            MySqlConnection connection = new MySqlConnection(connectionStr);
            connection.Open();

            string sqlStr = "SELECT * FROM fix_message";
            sqlStr += " WHERE id = " + input;

            MySqlCommand sqlCmd = new MySqlCommand(sqlStr, connection);

            // Use SqlDataAdapter and Dataset
```

```
            MySqlDataAdapter adapter = new MySqlDataAdapter(sqlCmd);
            DataSet dataSet = new DataSet();
            adapter.Fill(dataSet, "Queue");

            // id[0] | beginstring[1] | msgtype[2] | sendercomid[3] |
sendersubid[4] |
            // targetcomid[5] | targetsubid[6] | inboundmsg[7] | outboundmsg[8]
| status[9]
            beginString =
dataSet.Tables["Queue"].Rows[0]["beginstring"].ToString();
            msgType = dataSet.Tables["Queue"].Rows[0]["msgtype"].ToString();
            senderComID =
dataSet.Tables["Queue"].Rows[0]["sendercomid"].ToString();
            senderSubID = dataSet.Tables["Queue"].Rows[0][4].ToString();
            targetComID = dataSet.Tables["Queue"].Rows[0][5].ToString();
            targetSubID = dataSet.Tables["Queue"].Rows[0][6].ToString();
            xmlData = dataSet.Tables["Queue"].Rows[0][7].ToString();
            inboundMsg = dataSet.Tables["Queue"].Rows[0][8].ToString();
            outboundMsg = dataSet.Tables["Queue"].Rows[0][9].ToString();
            int status = (int)dataSet.Tables["Queue"].Rows[0][10];

            // Remove the record which is session not found and
            string sqlStr2 = "DELETE FROM fix_message WHERE id =" + input;
            MySqlCommand sqlCmd2 = new MySqlCommand(sqlStr2, connection);
            sqlCmd2.ExecuteNonQuery();

            // insert into the lastest row
            string sqlStr3 = "INSERT INTO fix_message(beginstring, msgtype,
sendercomid, sendersubid,";
            sqlStr3 += " targetcomid, targetsubid, xmlData, inboundmsg,
outboundmsg, status)";
            sqlStr3 += " VALUES('" + beginString + "', '" + msgType + "', '" +
senderComID + "',";
            sqlStr3 += " '" + senderSubID + "', '" + targetComID + "', '" +
targetSubID + "',";
            sqlStr3 += " '" + xmlData + "', '" + inboundMsg + "', '" +
outboundMsg + "', " + status + ")";

            MySqlCommand sqlCmd3 = new MySqlCommand(sqlStr3, connection);
            sqlCmd3.ExecuteNonQuery();

            // Console.WriteLine("Moved.");
        }

        public void OnCreate(SessionID sessionID)
        {
            MySqlConnection connection = new MySqlConnection(connectionStr);
            connection.Open();
            string sqlStr = "DELETE FROM session";
            MySqlCommand sqlCmd = new MySqlCommand(sqlStr, connection);
            sqlCmd.ExecuteNonQuery();

            Console.WriteLine("On Create : " + sessionID.ToString());
            this.sessionID = sessionID;
        }

        public void OnLogon(SessionID sessionID)
        {
            beginString = sessionID.BeginString;
            senderComID = sessionID.SenderCompID;
            senderSubID = sessionID.SenderSubID;
            targetComID = sessionID.TargetCompID;
```

```
            targetSubID = sessionID.TargetSubID;

            MySqlConnection connection = new MySqlConnection(connectionStr);
            connection.Open();
            string sqlStr = "INSERT INTO session(beginstring, sendercomid,
sendersubid,";
            sqlStr += " targetcomid, targetsubid)";
            sqlStr += " VALUES('" + beginString + "', '" + senderComID + "',";
            sqlStr += " '" + senderSubID + "', '" + targetComID + "', '" +
targetSubID + "')";
            MySqlCommand sqlCmd = new MySqlCommand(sqlStr, connection);
            sqlCmd.ExecuteNonQuery();

            Console.WriteLine("On Logon : " + sessionID);
        }

        public void OnLogout(SessionID sessionID)
        {
            beginString = sessionID.BeginString;
            senderComID = sessionID.SenderCompID;
            senderSubID = sessionID.SenderSubID;
            targetComID = sessionID.TargetCompID;
            targetSubID = sessionID.TargetSubID;

            MySqlConnection connection = new MySqlConnection(connectionStr);
            connection.Open();
            string sqlStr = "DELETE FROM session WHERE sendercomid = '" +
senderComID + "' AND";
            sqlStr += " sendersubid = '" + senderSubID + "' AND";
            sqlStr += " targetcomid = '" + targetComID + "' AND";
            sqlStr += " targetsubid = '" + targetSubID + "'";
            MySqlCommand sqlCmd = new MySqlCommand(sqlStr, connection);
            sqlCmd.ExecuteNonQuery();

            Console.WriteLine("On logout : " + sessionID);
        }

        public void ToAdmin(Message message, SessionID sessionID)
        {
            Console.WriteLine("To admin : " + sessionID + "\n" + message);
        }

        public void FromAdmin(Message message, SessionID sessionID)
        {
            // Check MsgType[35]
            msgType = message.Header.GetString(35);
            if (msgType.Equals(MsgType.XML_NON_FIX))
            {
                beginString = message.Header.GetField(8);
                senderComID = message.Header.GetField(49);
                senderSubID = message.Header.GetField(50);
                targetComID = message.Header.GetField(56);
                targetSubID = message.Header.GetField(57);
                xmlData = message.Header.GetField(213);
                // string xmlDataUTF8 = Base64ToString(xmlData);

                MySqlConnection connection = new
MySqlConnection(connectionStr);
                connection.Open();
                string sqlStr = "INSERT INTO fix_message(beginstring,
msgtype,";
```

```
                sqlStr += " sendercomid, sendersubid, targetcomid, targetsubid,
xmldata, inboundmsg)";
                sqlStr += " VALUES('" + beginString + "', '" + msgType + "', '"
+ senderComID + "',";
                sqlStr += " '" + senderSubID + "', '" + targetComID + "', '" +
targetSubID + "',";
                sqlStr += " '" + xmlData + "', '" +
message.ToString().Replace((char)1, '^') + "')";
                MySqlCommand sqlCmd = new MySqlCommand(sqlStr, connection);
                sqlCmd.ExecuteNonQuery();
            }
            Console.WriteLine("From admin : " + sessionID.TargetCompID + "/" +
sessionID.TargetSubID + "\n"
                + message);
        }

        public void ToApp(Message message, SessionID sessionId)
        {
            Console.WriteLine("To app : " + sessionID.ToString() + "\n" +
message.ToString());
        }

        public void FromApp(Message message, SessionID sessionID)
        {
            Console.WriteLine("From app : " + sessionID.ToString() + "\n" +
message.ToString());
        }
    }
}
```

FIX Application ของ Client ที่เขียนด้วย Java

ไฟล์ FIXInitiator.java

```
import java.io.*;
import quickfix.*;

public class FIXInitiator
{
       public static void main(String[] args)
       {
        try
        {
              String configFile = "config/initiator.cfg";
            SessionSettings sessionSettings = new SessionSettings(new
FileInputStream(configFile));
            FileStoreFactory storeFactory = new
FileStoreFactory(sessionSettings);
            LogFactory logFactory = new FileLogFactory(sessionSettings);
            MessageFactory messageFactory = new DefaultMessageFactory();
            InitiatorApp initiatorApp = new InitiatorApp(sessionSettings);
            Initiator initiator = new SocketInitiator(initiatorApp,
storeFactory, sessionSettings, logFactory, messageFactory);
            initiator.start();
            System.in.read();
              initiator.stop();
              }
        catch (Exception e)
        {
```

```
            System.out.println(e.toString());
        }
       }
}
```

ไฟล์ InitiatorApp.java

```
import quickfix.*;

public class InitiatorApp implements Application
{
    private SessionID sessionID = null;
    private SessionSettings sessionSettings = null;

      public InitiatorApp(SessionSettings sessionSettings)
      {
            this.sessionSettings = sessionSettings;
      }

       @Override
       public void fromAdmin(Message message, SessionID sessionID)
       {
             System.out.println("From admin : " + sessionID + "\n" + message);
       }

       @Override
       public void fromApp(Message message, SessionID sessionID)
       {
             System.out.println("From app : " + sessionID + "\n" + message);
       }

       @Override
       public void onCreate(SessionID sessionID)
       {
             System.out.println("On create : " + sessionID);
             this.sessionID = sessionID;
       }

       @Override
       public void onLogon(SessionID sessionID)
       {
             System.out.println("On logon : " + sessionID);
             try
             {
                   Message message = new Message();
                   message.getHeader().setField(new
quickfix.field.MsgType("n"));
                   String xmlData = "<Request><MsgCd>1111</MsgCd>" +
                         "<RqId>2222</RqId></Request>";
                   message.setInt(212, xmlData.length());
                   message.setString(213, xmlData);

                   Session.sendToTarget(message, sessionID);
             }
             catch (SessionNotFound e)
```

```
		{
			System.out.print(e.toString());
		}
	}

	@Override
	public void onLogout(SessionID sessionID)
	{
		System.out.println("On logout : " + sessionID);
	}

	@Override
	public void toAdmin(Message message, SessionID sessionID)
	{
		System.out.println("To admin : " + sessionID + "\n" + message);

	}

	@Override
	public void toApp(Message message, SessionID sessionId)
	{
		System.out.println("To app : " + sessionId + "\n" + message);
	}
}
```

ประวัติผู้จัดทำโครงงาน

ชื่อ – สกุล นายวิชุนันท์ เฉลิมเมือง

วัน เดือน ปีเกิด 26 กันยายน 2534

ประวัติการศึกษา

ระดับประถมศึกษา ประถมศึกษาตอนปลาย พ.ศ. 2546

โรงเรียนจงฝามูลนิธิ

ระดับมัธยมศึกษา มัธยมศึกษาตอนปลาย พ.ศ. 2552

โรงรัยนสิริรัตนาธร

ระดับอุดมศึกษา คณะเทคโนโลยีสารสนเทศ สาขาระบบสารสนเทสทางธุรกิจ พ.ศ. 2556

สถาบันเทคโนโลยีไทย – ญี่ปุ่น

ทุนการศึกษา - ไม่มี -

ประวัติการฝึกอบรม 1. โครงการ SET Internship Academy 2013

2. SAP ABAP Training

ผลงานที่ได้รับการตีพิมพ์ - ไม่มี -