Name: Yuen Chi Yuen, Jeffrey

Student ID: 97488725

Supervisor: Dr Chun, Andy H W (HWC)

Project Assessor: Dr WU, ANGUS

HWC-01-99A-BSIT

Using Message-Oriented Middleware To Develop Resilient Web-based Applications

Table of Contents

PART 1 OVERVIEW 4

1.1 Project Abstract 4

1.2 Project Aims 4

1.3 What is Message Oriented Middleware (MOM) 4

1.4 The Real System, Hong Kong Stock Exchange AMS/3 7

PART 2 MS MESSAGE QUEUE SERVER 8

2.1 The advantages of employ MSMQ in this project 8

PART 3 STOCK TRADING 9

3.1 Overview 9

3.2 Market Order/Limit Order 9

3.3 Change Order/Canceled order 9

3.4 Transaction Status 10

3.5 Broker Windows 10

3.6 System Design 11

3.7 Selected Source Code 16

PART 4 STOCK QUOTATION 19

4.1 Overview 19

4.2 Stock History Chart by ASP and Excel and real time quote 19

4.3 Stock History by Java Applet and RMI 19

4.4 System Designs 20

4.4 User Interface 23

4.5 Selected source codes 24

PART 5 CLIENT SERVICE 28

5.1 Overview 28

5.2 Transaction Statement 28

5.3 Account Information 28

5.4 WatchList 29

5.5 Personal Information 29

5.6 System Design 29

5.7 User Interface 30

5.8 Selected Source Code 31

PART 6 DATABASE DESIGN 34

PART7 FURTHER DEVELOPMENT 35

7.1 Interface connection for AMS/3 OG 35

7.2 Integrate with Microsoft Transaction Server 35

7.3 Real time push Stock Data by Callback 35

CONCLUSION 36

REFERENCE: 37

PART 1 Overview

1.1 Project Abstract

Electronic trading in the Internet is the next trading model in the coming century. E- trading via the Internet not only convenient for customer but also save the operation cost for whole trading process, that is advantage both of customer and service provider. To practice e-Commerce models, a reliable and completes IT infrastructure is the key to success. In this project, a **Web-enable Stock Trading System** was implemented. It shows how the latest IT technology to provide a reliable and complete solution for E-Commerce over the Internet. This project divided into two parts. First Part is Stock Trading (Buy/Sell) system by employed the **MS Message Queue Server**, to provide a reliable order and asynchronies process. Second part is Stock Price Quoting System; **Java RMI** technology will demonstrate the real client-server Architecture over the Internet.

1.2 Project Aims

- To investigate the architecture combining Internet and Middleware Technologies
- To develop a web enable stock trading and stock price quote system
- By MSMQ, Java RMI and ASP, Take advantage of those service in e-commerce

1.3 What is Message Oriented Middleware (MOM)

Middleware is layer of software situated between the operating system and the application, allowing them to exchange information. **The main goal of middleware is to solve the integration of software.** Now a day, computer system is developing in an extremely fast rate. Some organization may have different computer system and application that was implemented in different platform. The communication between those application was directly connected and as they needed. Finally, the system will end up with an inextricable system of inter-application connection. Thus, a new architecture was suggested to integrate the distributed application in a common unique communication interface-middleware.

The benefit from middleware

1. **Application Integration**

   It provided a standardized system-independent communication interface for applications.

2. **Reliable data transfer**

   The delivery of message was assured even in system crash

3. **Adaptation to communication traffic**

The bandwidth of communication bus is able to sustain an increase in traffic due to the addition of applications. Middleware is able to adapt to change since it forms every application system skeleton.

4. Diversity of communication structure

Application connected by middleware, they can communicate one-to-one or one-to-many by send the message to communication bus once.

5. Logical communication interface

The communication between applications is able to use a logical representation, such as name, not physical address again.

6. Transaction Support

Sometime, we need to use different application to complete a single transaction. By introduce middleware, it give out a single communication interface for system. The transaction is never committed unless all operation was completed. Also middleware allow transaction rollback in cause abort transaction.

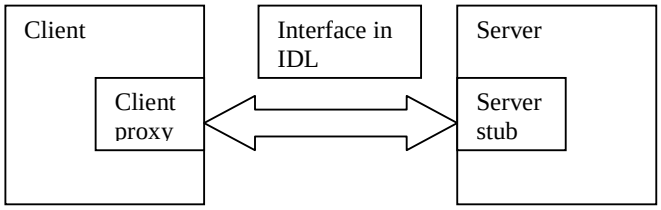
Now a day, there are three main middleware technology exist.

- Message-based middleware

  Applications communicate together by message exchange. It provided a *reliable*, *asynchronous* communication and data exchange for applications.

- RPC-based middleware

  Client and Server communicate by using standardized IDL (Interface definition Language). Thus Client and Server is programming language independent. The client call the remote procedure as local procedure calls by the service provided by RPC-middleware.



- Distributed-object based middleware

  This model is similar to RPC, but not procedure-oriented, that is Object Oriented. An Object oriented approach to for application communication. Objects all distributed across network, client object can execute an server object across the network by the Object Request Broker (object middleware). CORBA IDL (defined by Object Management Group) and DCOM (defined by Microsoft) was widely used standard in this model.

In this Project, the Message Oriented middleware will be investigated.

Message Oriented middleware (MOM)

As mention above message oriented middleware (MOM) enable applications to communicate with queuing middleware over network via a series of messaged that are stored in queue while they are waiting delivery. In each application, there are two queue for communication, one is sent queue another is receipt queue. Beside the benefited inherited from middleware. There are two main principle functions by MOM.

Synchronous and Asynchronous communication

Application can use MOM to send messages and continue processing regardless of whether the receiving application is running or reachable (network down or mobile user) over the network.

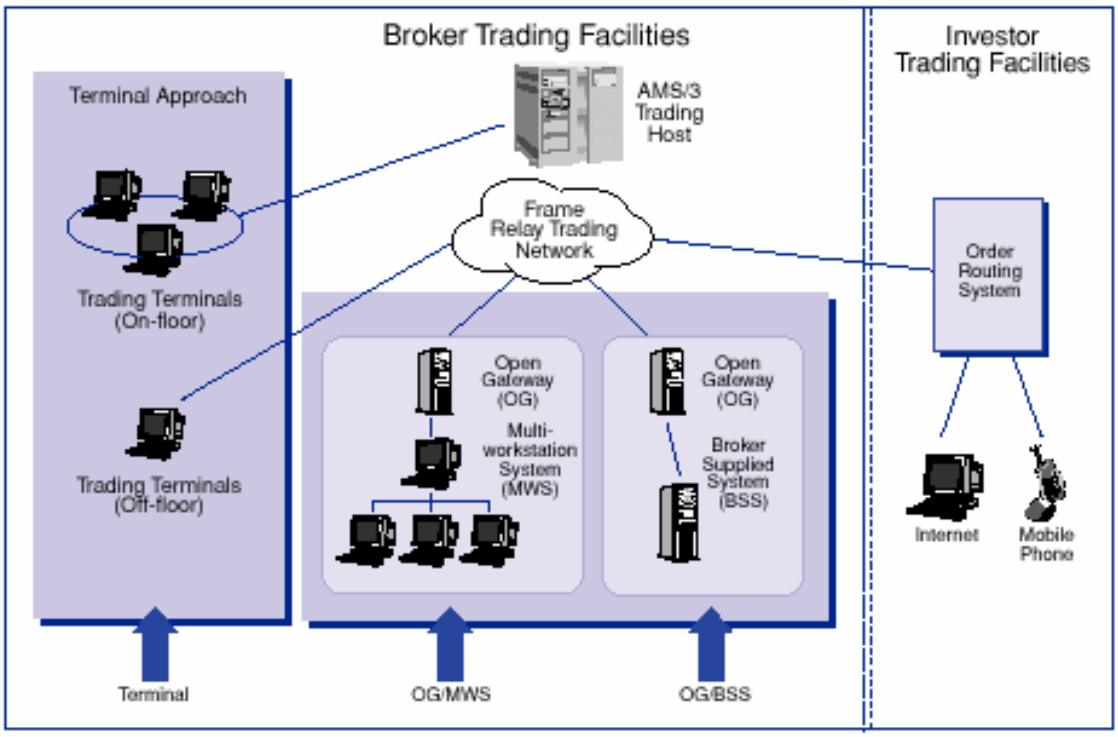
Message Delivery Guarantee (one time in-order Delivery)

By backing up the message in disk and the log-base recovery technique to protest against the unreliable network communication or system failure. By the control provide by MOM, the same message if guarantee to delivered exactly one time. It is essence for e-trading application for no repeating order.

1.4 The Real System, Hong Kong Stock Exchange AMS/3

In the middle of year 2000, the Hong Kong Stock Exchange will release the new stock trading system. Some new trading method will introduce in to this system, such as single price auction and market making are target to support future market development needs and facilitate the launching of new investment products. Another new feature is providing an

The AMS /3 System Overview



Open Gateway (OG) and **Order Routing System (ORS).** ORS enables investor to input trading requests electronically, such as Internet, Mobile phone. The trading request will automatically routed to broker for approval and submission to the market for matching and trade generation. This will be a **Sun Solaris Enterprise Server** of connect to AMS/3 by TCP/IP with Message Tag and secured with SSL.

Actually, this project's system would work like ORS and work behind OG as the broker properties, provide an interface for investor to do stock trading. This project's system is one of solution to provide a online stock trading service, and embedded different middleware technology, such as Message Middleware and JAVA RMI.

2.1 The advantages of employ MSMQ in this project

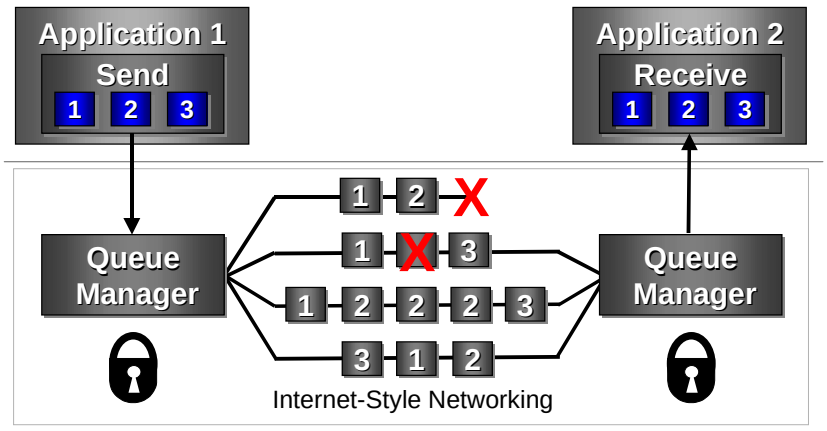
The MS Message Queue Server is the new component come with the Windows NT option pack. It provides loosely coupled and reliable network communication service based on a message model.

Stockbroker need applications that can accept trade request from customer and deliver the quickly, reliable, and in the order they were taken to back-office application that process the order. Lost order or orders that are delivered more than one time by mistake can have a serious consequence. With MSMQ, order entry application can send trade request using transactions delivery mode in MSMQ. That is promise that the delivery of order message will **in order and exactly one time**. In the event that machine or network failures occur, MSMQ will recover all message automatically- preventing any message loss**. (Journalized Communication)**

On the other hand, by COM component interfaces in MSMQ and Active Server Page scripts, an MSMQ messages from ASP scripts. When the user of a browser-based application accesses an ASP that sends MSMQ messages (Orders), Microsoft Message Queue Server will return control to the ASP quickly, and the user will not need to wait for the messages to be processed by the message's receiver. Because the processing performed by receivers can be complex, time-consuming, and sometimes performed during off-hours, using MSMQ enables to build Web-based applications that remain *available* and *responsive* to Clients. Also, because ASP scripts can send requests even to back-end applications that are not running or have failed, browser users perceive MSMQ-enabled Web sites to be highly available.

Moreover, several broker site could be setup and different MSMQ site and routing server can set up for different kind of orders. In case one of the broker sites was crashed, the message order is able route to other available site to serve the request. This type of configuration could achieve **load balancing**.

Reliable communication under unreliable network



PART 3 Stock Trading

3.1 Overview

In this System, user able to placing orders to buy and sell over the Internet. Moreover, user able to cancel and modify the order before the transaction completed. For illustrate the function of MSMQ, a Broker Window was implemented to view and authorize all orders. However, the broker window can replace by an automatic validation system and constructed with AMS/3 OG Compatible message format for the real life system.

3.2 Market Order/Limit Order

In real market, there are two type of order. First is Market Order, second is Limit Order. For market order, that means the order could only been place and execute a specific price range, otherwise the market will reject the order. For limited order, the order could only been executed in the market at the specific price range within the assigned period. Such as targeted stock price is 10 dollar right now, an order can be place at price 9 dollar and expire date is tomorrow 4:00 p.m. The order will execute, if the stock price drop to 9 dollar before tomorrow 4:00p.m. Limit Order is useful for investor to plan their investment strategy, such as Sell Stop Loss Order, Buy Stop Loss Order and Sell Stop Earn Order.
Procedure to placing order:

1. Check the BUY/SELL Box
2. Enter Stock Code
3. Enter Quantity
4. Enter the Expire order time (For Limit Order Only)
5. Enter trading Password
6. Press Execute

For each order, client will receipt e-mail from system; the email included the trading information and order number for reference. The email will construct by the **CDONTS.NEWMAIL** Object that come form SMTP server provided from IIS 4.0.

3.3 Change Order/Canceled order

Before the Order Executed, all orders are able to be change and cancel. And a Change or Cancel message will send to broker for further process. For change order, user can change the target price and quantity.

3.4 Transaction Status

Generally speaking, after an order placed, an order will have following status, user able to view the placed order transaction status by web browser.

- **Sending** - sending orders, the order was sending to market
- **Queuing** - orders is queuing, the order was sent to market, waiting to match price
- **Executed** – the order was deal
- **Rejected** - rejected by market (AMS), due to order price out of range.
- **Waiting Approval –** waiting for broker to authorize the order (for illustration in this system)
- **Approved –** the order was Approved by the broker (for illustration in this system)

3.5 Broker Windows

A Broker Window was implemented to view and authorizes all orders through browser. First it will **peek** the entire message for a queue, after authorize the order, the message will treated as **received**. The different between **peek** function and **receive** function is that peek only read the message from the queue but and not delete it and receive will read and delete it form the queue.

Functions of Broker Windows:

1. Retrieve and print all message from MSMQ in HTML format.
2. Able to view different Order Queue
3. Authorize Orders
4. Rejected Orders

3.6 System Design

3.6.1 System Background

Language: ASP, VB Script, and JAVA Script

Platform: MS WINNT Server 4.0, IIS4.0

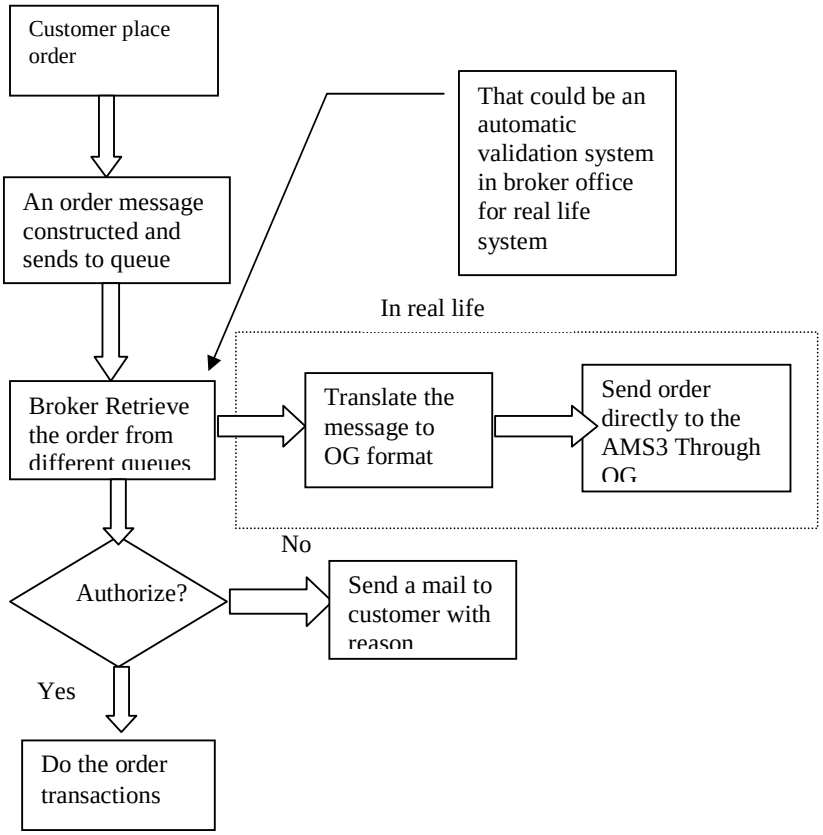
Database: MS SQL Server 6.5

Middleware: MS Message Queue Server 1.0

3.6.2 System Overview

In real life and come with AMS/3, all trading order able collected from different channel. And all orders are automatic send to market without any manual input or process. However, in this project, for illustrate the feature of middleware, a Broker windows was implemented to authorize and view all order by web browser. And to simplify the trading process (No Bid and Ask matching process), if the broker approved the order, it will be assume that the transaction was completed.

The whole trading process as below:



Message Definitions

By MSMQ, all the order will pack as a message and send to different kind of order queues.

Example:
A Customer place a buy order her/his **Customer ID** is 1 and **order ID** B30252, **stock code** is 5, **price** is 4.3 and **quantity** is 5000.

The message body will as below:

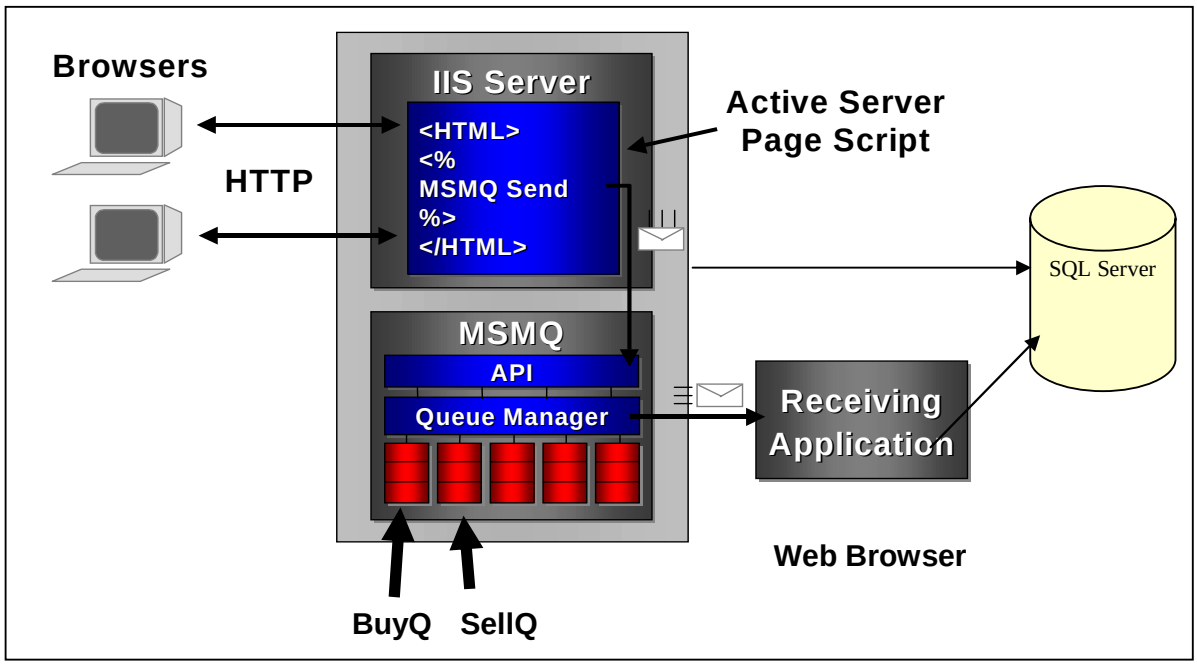
```
UserID=1&
OrderID=B30252&
stock_id=5&
price=4.3&
qty=5000&
total=21500
```

Queues Definitions

There are following queues was defined in MSMQ server for different kinds of orders

- **Buy**
- **Sell**
- **Limit Buy**
- **Limit Sell**
- **Cancel**
- **Change**

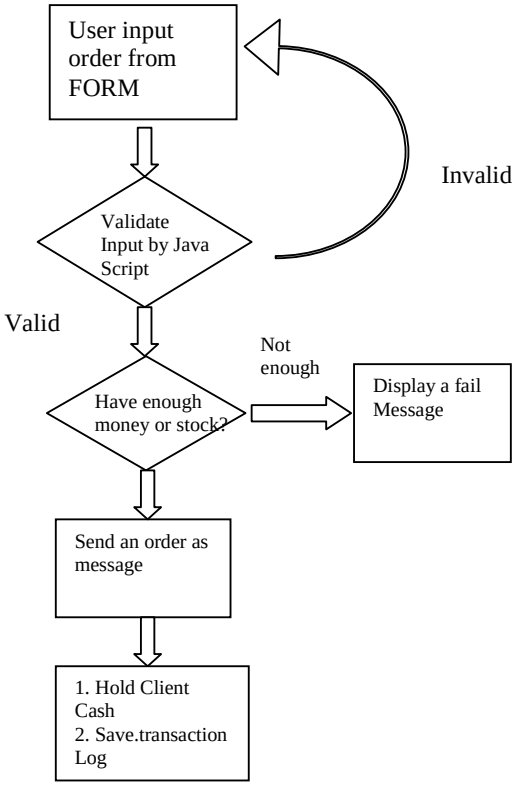
3.6.3 System Architecture



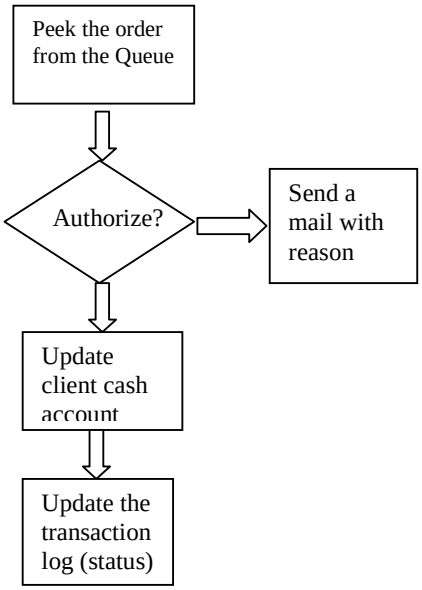
This is a three-tier architecture. Client able place and check order by web browser and broker able to retrieve the order from MQ server the save the transaction to database. The third tier is the SQL server; it stored the client's cash, stock account and the transactions log.

3.6.4 *Flowcharts*

Buy/Sell Order

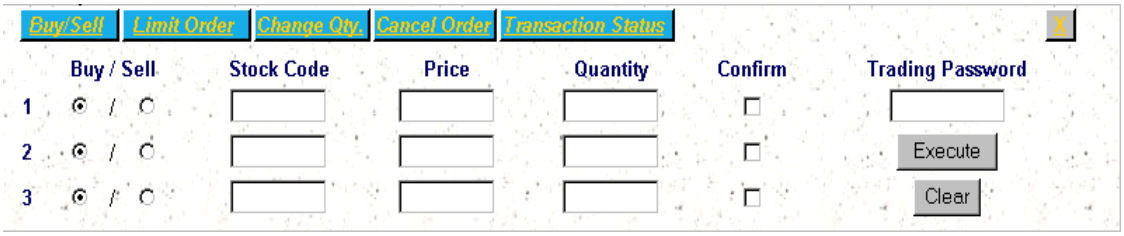


Broker Approval

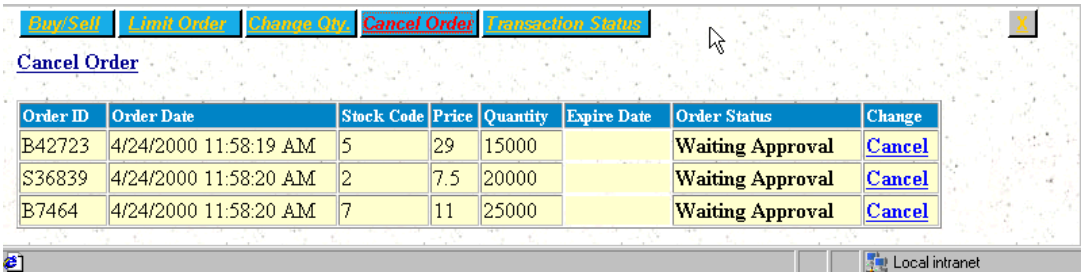


3.6.5 User Interface

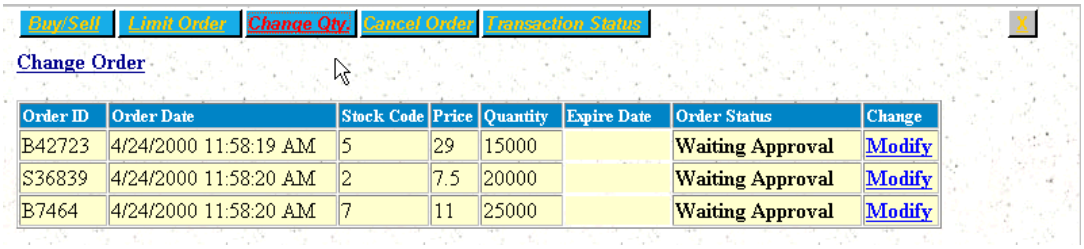
Client BUY/SELL Order Interface



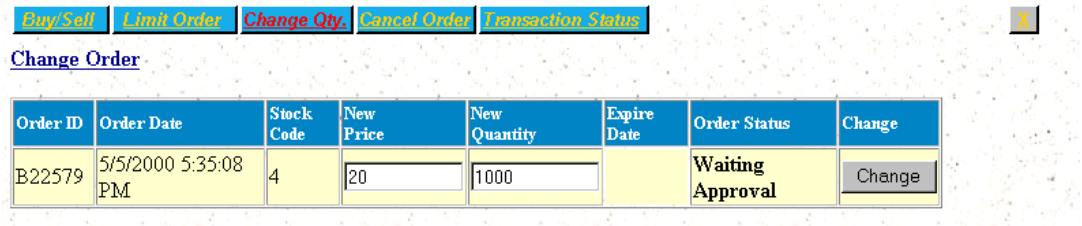
Cancel Order Interface



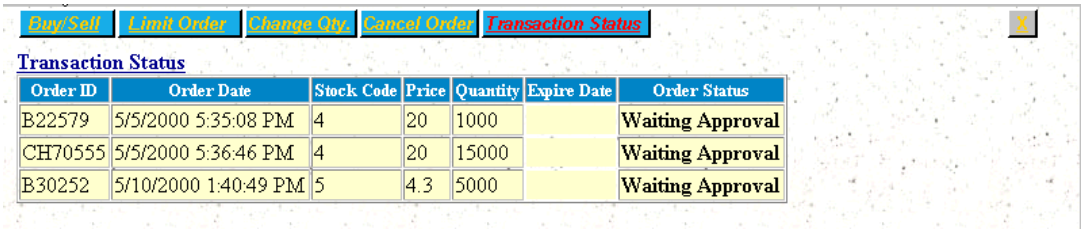
Cancel Order Interface



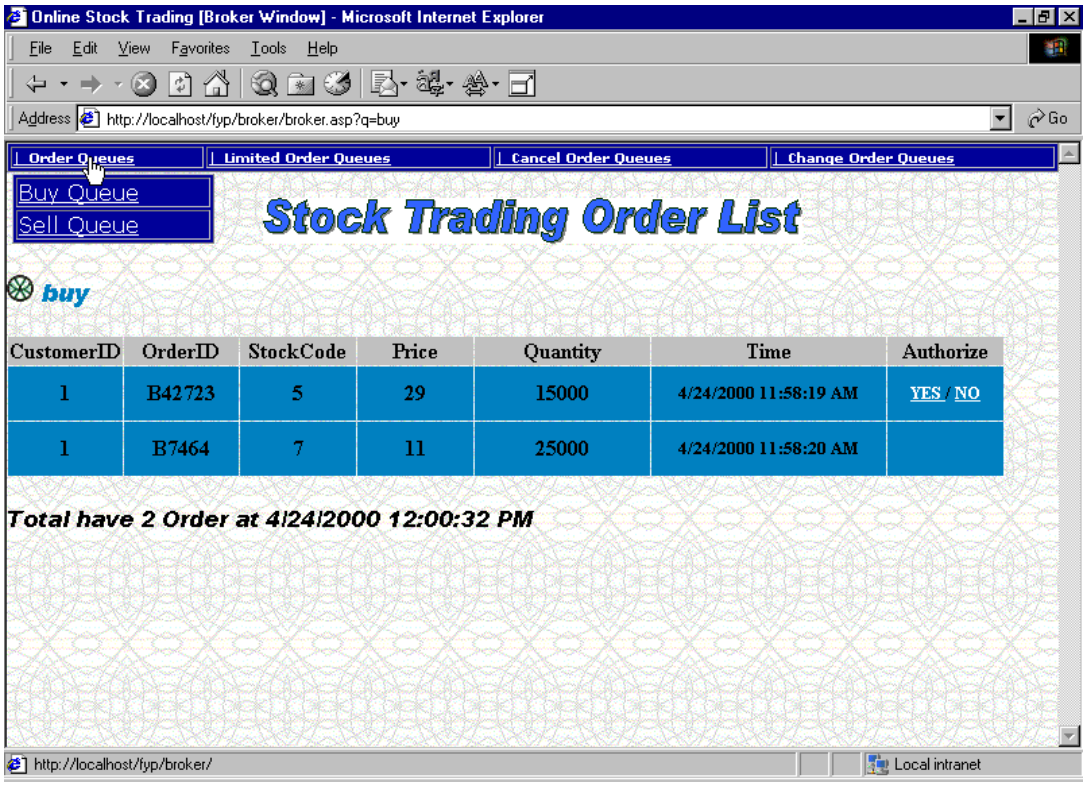
Change the order and send the change order by click Change



Transaction status interface



Broker Window



3.7 Selected Source Code

Order.asp

```
‘it send the message to a specificed Q
sub SendMessage(strQueue,strLabel, strBody)

    Dim mqQuery
    Dim mqQInfos
    Dim mqQInfo
    Dim mqTXmsg

    Set mqTXQueue = Server.CreateObject("MSMQ.MSMQQueue")
    Set mqQuery   = Server.CreateObject("MSMQ.MSMQQuery")
    Set mqQInfos  = Server.CreateObject("MSMQ.MSMQQueueInfos")
    Set mqTXmsg   = Server.CreateObject("MSMQ.MSMQMessage")

      Set mqQInfos = mqQuery.LookupQueue(,,strQueue)
    mqQInfos.Reset
    Set mqQInfo = mqQInfos.Next
    Set mqTXQueue = mqQInfo.Open(2,0)

    'Open the queue with Send access.
    'Create a new message object
    'Set the body and label properties
    mqTXmsg.Body = strbody
    mqTXmsg.Label = strLabel
    'Send the message
    mqTXmsg.Send mqTXQueue, false
    'clean up
    mqTXQueue.Close
    set mqTXmsg=Nothing
    set mqTXQueue=Nothing
    set mqQInfos=Nothing
    set mqQInfo=Nothing
End Sub

……….

……….
strOrderID = strHead & CStr(Int((100000 * Rnd) + 1))
OrderList(OrderNo,assignOrderID)=strOrderID
OrderList(OrderNo,OrderTime)= Now
oneOrderCash=OrderList(OrderNo,price)*OrderList(OrderNo,Qty)
'Define the order message body and send the message
Dim strBody
strBody="UserID="&strUserID&"&"
strBody= strBody & "OrderID="&strOrderID&"&"
strBody= strBody & "stock_id=" & Cstr(OrderList(OrderNo,stkcode))&"&"
strBody= strBody & "price=" & Cstr(OrderList(OrderNo,price))&"&"
strBody= strBody & "qty=" & Cstr(OrderList(OrderNo,Qty))&"&"
strBody= strBody & "total=" & Cstr(oneOrderCash)&""
SendMessage CStr(OrderList(OrderNo,bos)), strUserID, strBody
```

Transcation_status.asp

```
<%CustomerID=Session("USERID")

strDbConnection="DATABASE=stock;DSN=sql_stock;UID=stock_client;Password=;"
SqlGetWatchList="SELECT * FROM transcations where
CustomerID="&CustomerID&" AND (OrderStatus='Approved' OR
OrderStatus='Waiting Approval')"
Set Con=Server.CreateObject("ADODB.Connection")
Con.Open strDbConnection
Set TranRS=Con.Execute(SqlGetWatchList)
%>
<html>
<TABLE border=1 cellPadding=1 cellSpacing=1 width="75%">
  <TR>
    <TD align="center" bgcolor="#0080C0"><font size="2"
    ……
    ……
    <TD align="center" bgcolor="#0080C0"><font size="2"
color="#FFFFFF"><b>Order Status</b></font></TD></TR>
  <TR>
  <%
       DO While Not TranRS.EOF
           'get a transcation records
            OrderId=TranRS.Fields("OrderId")
            OrderDate=TranRS.Fields("OrderDate")
            StockCode=TranRS.Fields("StockCode")
            price=TranRS.Fields("price")
            qty=TranRS.Fields("Quantity")
            expdate=TranRS.Fields("ExpireDate")
            status=TranRS.Fields("OrderStatus")
  %>
    <TD bgcolor="lemonchiffon"><%=OrderID%></TD>
    <TD bgcolor="lemonchiffon"><%=OrderDate%></TD>
    <TD bgcolor="lemonchiffon"><%=StockCode%></TD>
    <TD bgcolor="lemonchiffon"><%=price%></TD>
    <TD bgcolor="lemonchiffon"><%=qty%></TD>
    <TD bgcolor="lemonchiffon"><%=expdate%></TD>
     <TD bgcolor="lemonchiffon"><B><%=status%></B></TD>
    <%
    TranRS.MoveNext%>
    </TR>
    <%loop%>
  <%
   'clear
   Con.Close()
   SET TranRS=Nothing
   %>
</TABLE>
</body>
</html>
```

broker.asp

```
‘This script retrive the message for MSMQ and show order as a table for
‘broker to authorize
    Q=Request.Querystring("q")
    Response.Write(Q)
    set BuyQuery = Server.CreateObject("MSMQ.MSMQQuery")
    Set BuyQinfos = BuyQuery.LookupQueue (,,Q)
    'point to the first queue in the collection
    Set BuyQinfo = BuyQinfos.Next
    'open the queue with peek access level
    Set BuyQ = BuyQinfo.Open(32, 0)
    'get the first message in the queue, but not receive it yet
    Set BuyQMsg = BuyQ.PeekCurrent(,,100)

    if BuyQMsg IS Nothing Then%>  Queue  NO More Order <%else%> </P>
<table border="0" cellPadding="1" cellSpacing="1" width="759"
 style   ="WIDTH: 740px" id="TABLE1" height="55">
  <tr>
    <th align="middle" width="10%" bgcolor="#c0c0c0" height="7"><font
color="#000000">
     ……
     ……
     ……
    <th align="middle" width="15%" bgcolor="#c0c0c0" height="7"><font
color="#000000" face="Times New Roman" size="3">Quantity</font></th>
    <% IF Q="limit_buy" OR Q="limit_sell" Then%>
      Expire Time</th>
    <%
    end if
    %>
    <th align="middle" width="20%" bgcolor="#c0c0c0" height="7"><font
color="#000000" face="Times New Roman" size="3">Time</font></th>
    <th align="middle" width="10%" bgcolor="#c0c0c0" height="7"><font
color="#000000" face="Times New Roman" size="3">Authorize</font></th>
  </tr>
<%
       While Not BuyQMsg Is Nothing
                  NumOrder=NumOrder+1
                  strBody = BuyQMsg.Body
                  tlist=split(strbody,"&")
                  UserID=split(Cstr(tlist(0)),"=")
                  OrderID=split(Cstr(tlist(1)),"=")
                  StockID=split(Cstr(tlist(2)),"=")
                  price=split(Cstr(tlist(3)),"=")
                  qty=split(Cstr(tlist(4)),"=")

                  'Retrive the Expire time, if it is limie order
                  IF Q="limit_buy" OR Q="limit_sell" Then
                     ExpireDay=split(Cstr(tlist(5)),"=")
                     ExpireTime=split(Cstr(tlist(6)),"=")
                  End IF
                  time=Cstr(BuyQMsg.ArrivedTime)
                  set BuyQMsg=BuyQ.PeekNext(,,1000)
      %>
```

PART 4 Stock Quotation

4.1 Overview

Stock Real-Time Quote service is essential parts for a stock trading system. In this part, two type of stock chart will plot by different technologies. Nowadays in Hong Kong, most of financial chart provided by wed site, such as Quamnet.com, e-finet are query the result from database and export the stock chat GIF image. ASP and MS Excel 2000 Library implemented it in this project. Moreover, a real client server model stock chart Applet was also implemented for the investigation of Java RMI.

4.2 Stock History Chart by ASP and Excel and real time quote

In this page, two main functions will be done, first is get the real time information from database and show as HTML after user input a stock code. Second is plotting the history graph by EXECEL library and export as a GIF image in real-time. To connect Excel and ASP, an Excelchart.dll library was employed. This feature is very useful for real time graph, such as current day stock trends or minims update graph.

Functions of real time stock quote:

- List all the stock information

1. Nominal
2. Today High
3. Today Low
4. Bid Price
5. Ask Price
6. Change
7. Volume
8. Share Traded
9. P/E Ratio

- Query the history data, plot the stock chart and export as a GIF image.

Advantages of this Method and Component

- There is no demand from the client - the thinnest will do
- There is no need to know MS Excel in detail or the complex hierarchy of objects in it
- Flexibility to modify the current code of the component for further enhancements
- Depends on a powerful and popular tool like MS Excel

4.3 Stock History by Java Applet and RMI

In this part, a Java Applet Client and server was implemented. By the Applet client, user able to see different kinds of charts, BAR chart, Line Chart, and volume chart. Moreover, by click the specific date point of the chart, the respective daily records will show in the left hand side. The Java Applet Interface was implemented by Softbear Inc. and freely to modified and

use. Thus the data modeling, server connection and server was needed to implement in this project.

The function of Java Applet will as below:

1. Query the history data from Quote server
2. Plot the different kind of charts form the data provided (implemented by Softbear Inc, the mkmodel.class, mkview.class).

The function of Quote Server will as below:

1. Query stock history data from database
2. Provided an interface for client applet to query data
3. Pack the daily record as a collection for pass back to client applet (Sterilization)

Advantage of this method

- More function can don in client applet, such as different kind of chart
- User can check every daily record by click the chart
- No dependent software

4.4 System Designs

Stock History Chart by ASP and Excel

To provide an interface for ASP to access the Excel 2000 functions, an excelchat.dll component was implemented by Visual Basic.

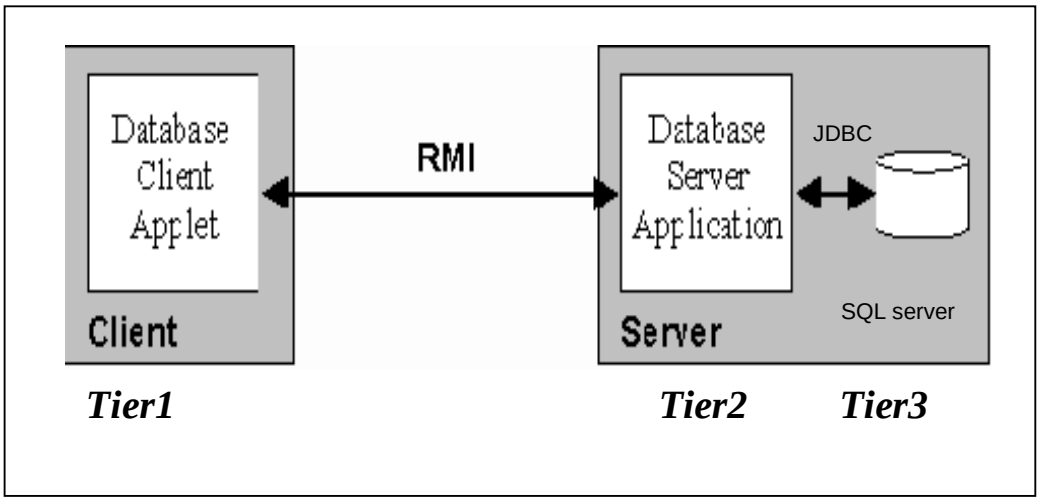
It provided the following functions:

`AddDataSeries`
It add a new set of data from a recordset column
`ExportToGif`
To export the chart to a specified GIF file
`Reset()`
To reset the current data and prepare for another chart
`SetBackgroundEffect`
To Set Special effect, such as 3Dchart
SetChartTitles
To set the chart title
SetChartWithDataProps
To set properties possible only after some data is added
SetChartOptions
To set the chart options like type size legend etc.
Procedure to export a chart:

1. Create the excelchart components
2. Get the history records set from database.
3. AddataSeries to excelchart.dll components
4. Set attributes of the chart
5. Export the chart as a GIF image

Stock History by Java Applet and RMI

System Architecture



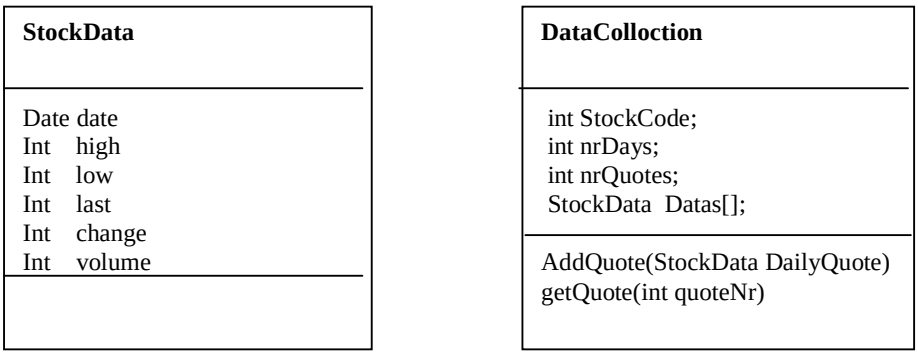
This is three-tier architecture, the communication between the client applet and DataQuote Server is Java RMI, and the server connects and queries the SQL server by JDBC/ODBC connection.

The interface file

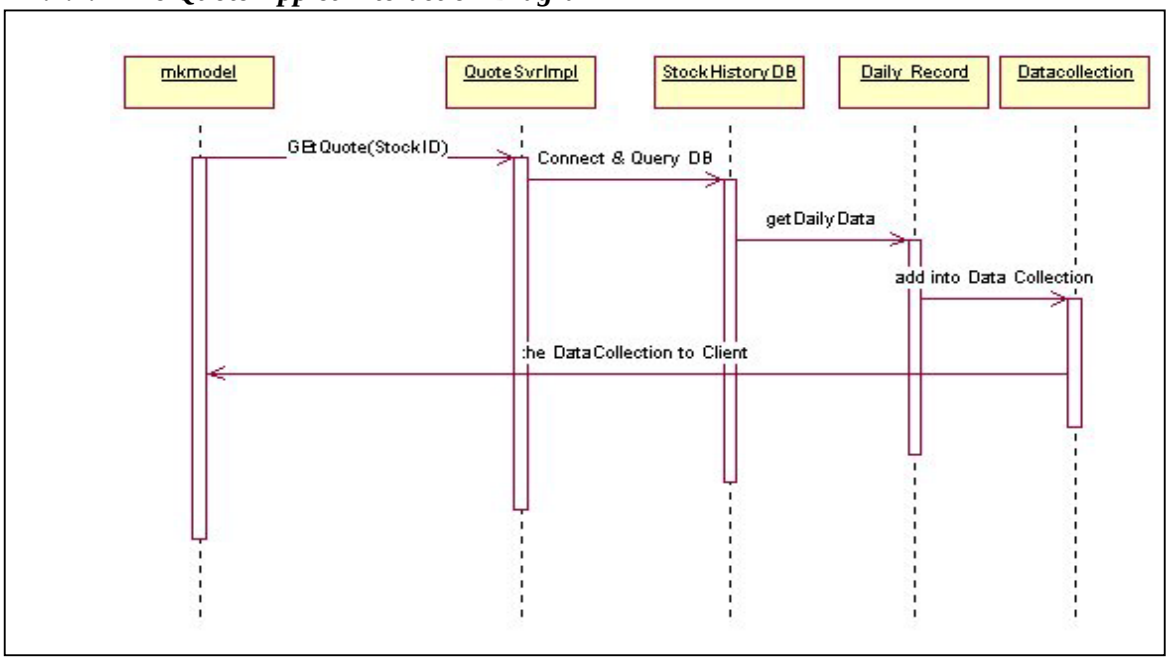
QuoteSvrInterface.java

```
public interface QuoteSvrInterface extends  java.rmi.Remote {

  public DataCollection GetQuote(int StockCode, int nrDays)
   throws java.sql.SQLException,java.rmi.RemoteException;

}
```

The Class diagram of the Stock Data (daily recods) and Datacolloction



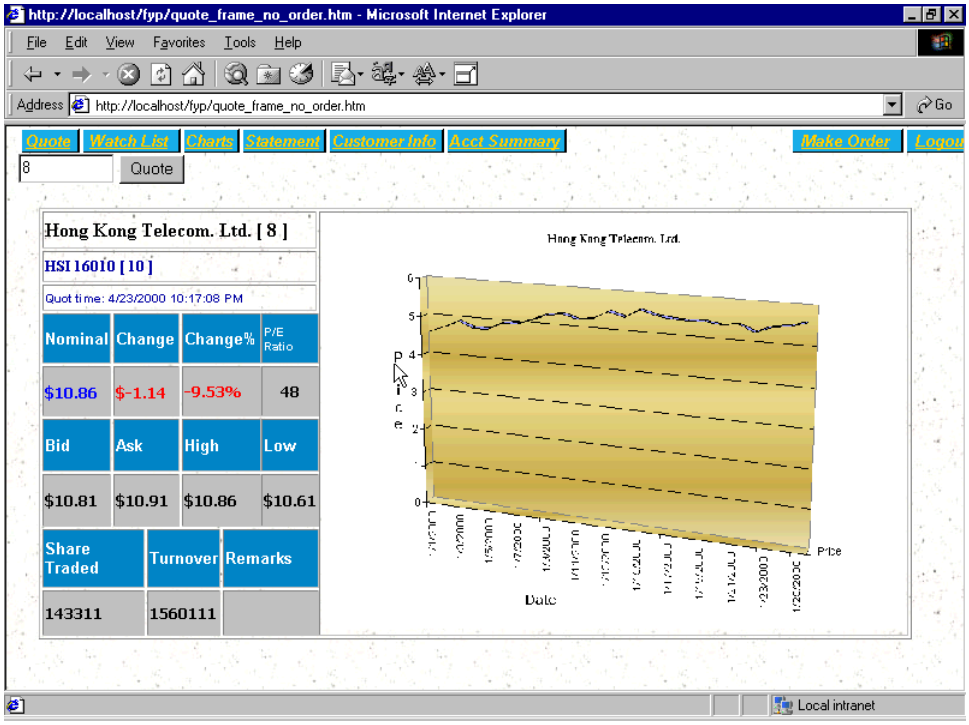
4.4.2.4 The Quote Applet Interaction Diagram



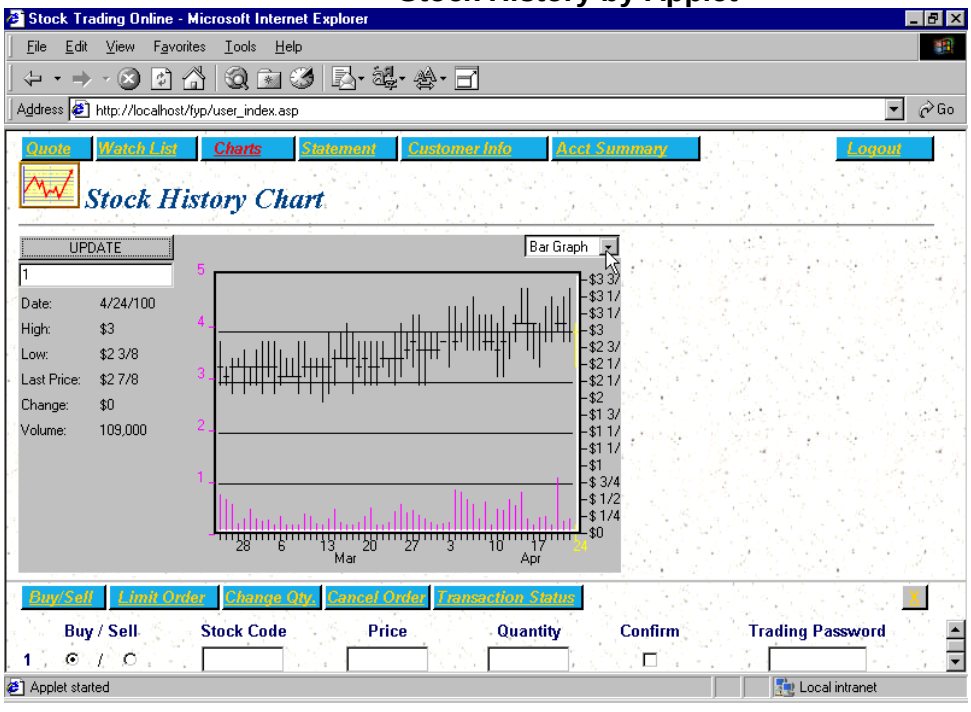
1. Once the client applet input the stock code and click update button, it will invoke the **GetQuote** (StockID) method.
2. The **QuoteServerImpl** object will create the **StockHistoryDB** object to query the history data.
3. StockHistroyDB get the daily records (2 month)day by day from database
4. Add daily record into the **Datacollection** object by AddQuote method.
5. Return the serialized data collection object to client applet.

4.4 User Interface

Real time stock quote and chart



Stock History by Applet



4.5 Selected source codes

Stock_Quote.asp for the EXCEL Chart

```
<%@ Language=VBScript %>
<BODY background="../img/romtextb.jpg" topmargin="0" leftmargin="10">
<% If Request.Form("StockCode") <> "" Then
intLastQuote=Request.Form("StockCode")

SqlGetRTData=             " SELECT r.*,c.CompanyName "
SqlGEtRTData=SqlGEtRTData+" FROM real_time_price AS r ,company AS c "
SqlGEtRTData=SqlGEtRTData+" WHERE r.StockCode="&intLastQuote&""
SqlGEtRTData=SqlGEtRTData+" AND r.StockCode=c.StockCode"
'Response.Write(SqlGEtRTData)
strDbConnection="DATABASE=stock;DSN=sql_stock;UID=stock_client;Password=;"
Set Con=Server.CreateObject("ADODB.Connection")
Con.Open strDbConnection

Set QuoteRS=Con.Execute(SqlGetRTData)

strName=QuoteRS.Fields("CompanyName")
floatNominal=QuoteRS.Fields("price")
intPE=QuoteRS.Fields("pe")
floatBid=QuoteRS.Fields("bid")
floatAsk=QuoteRS.Fields("ask")
floatHigh=QuoteRS.Fields("high")
floatLow=QuoteRS.Fields("low")
intShareTrd=QuoteRS.Fields("share_traded")
intVol=QuoteRS.Fields("turnover")
floatOpen=QuoteRS.Fields("last_close")
updatetime=QuoteRS.Fields("last_update")
intChange=floatNominal-floatOpen
intChangeP=(floatNominal-floatOpen)*100/floatOpen

%>
<P><!-- #INCLUDE FILE="ExcelChart.inc" --><%
  Dim oRs
  Dim oConnection
  Dim strQuery
  dim strSQLStkName
  dim StockCode
  dim strStartDate
  dim strEndDate
  dim NameRS
  dim RSCompanyName

   'StockCode=Request.QueryString("StockCode")
   StockCode=intLastQuote
   if StockCode="" then
      StockCode=1
   end if

   strStartDate="1/1/2000"
   strEndDate=DATE

  Set oConnection = Server.CreateObject("ADODB.Connection" )

  '--- open the connection
  oConnection.Open "Driver={SQL Server}; Server=(local); Database=stock;
UID=stock_client; PWD="

   '--- build the query string
```

```
 strQuery = "SELECT stock_code,last,date FROM History WHERE
stock_code="&StockCode&" AND date Between '"&strStartDate&"' and
'"&strEndDate&"'"
 strSQLStkName="SELECT CompanyName FROM Company Where
StockCode="&StockCode&""
 'strQuery="SELECT stock_code,last,date FROM History WHERE stock_code=2
AND date Between '1/1/2000' and '3/30/2000'"
 ' Response.Write(strQuery)
    '--- get the data in recordset
  Set oRs = oConnection.Execute( strQuery )
    '--- check if data is there
    if( oRs.EOF ) then
      Response.Write( "No data available" )
      Response.End
    end if
 Set NameRS =oConnection.Execute(strSQLStkName)
      RSCompanyName=NameRS.Fields("CompanyName")
'Response.Write(  RSCompanyName)
%>

<!-- Section 2: using component to build chart -->
<%
  Dim oExcelChart
  Dim strGIFFileName

  '--- create the object
  Set oExcelChart = Server.CreateObject( "ExcelChart.cExcelChart" )

  '--- pass the set of points to be put on chart
  oExcelChart.AddDataSeries oRs, "last", "Price", FALSE,FALSE
  '--- set the headings for x-axis
  oExcelChart.SetXAxisHeadings oRs, "Date"

  '--- provide chart and axis titles
  oExcelChart.SetChartTitles ""&RSCompanyName&"","Date","Price"
  '--- set chart type and plot area width and height

   oExcelChart.SetChartOptions  xl3DLine,5000,4000,false,true,true
    '--- set special background effect
  oExcelChart.SetBackgroundEffect msoGradientGold

  '--- prepare the GIF file name, target path must have write permission
  strGIFFileName = CStr( Server.MapPath(".") & "\"+
"EXCELCHART"+""&StockCode&""+".GIF" )
    '--- export the chart
  oExcelChart.ExportToGif( strGIFFileName )

  '--- clean up
  Set oExcelChart = nothing
  Set oRs = nothing
  Set oConnection  = nothing
%>
```

The **DBQuote** method from Java Applet client **mkmodel.java** for query data from database by Java RMI

```
private void DBQuotes(int StkCode, int nrDays) throws
java.lang.Exception,java.sql.SQLException {

   QuoteSvrInterface Server;
   int index=0;
   float high;
   float low;
   float last;
   float change;
   int vol=0;
   int ihigh=0, ilow=0, ilast=0, ichange=0;
   java.util.Date rday;

   DataCollection records= new DataCollection(StkCode,nrDays);
   StockData DailyData;
   Server=(QuoteSvrInterface) Naming.lookup(rmiServer);
   //Get Quote Data Object by RMI
   records=Server.GetQuote(StkCode,nrDays);
   for (index=0; index<nrDays; index++){
        QuoteModel quoteModel = new QuoteModel();
        DailyData=records.getQuote(index);
        rday=  DailyData.getDate();
        high=  DailyData.getHigh();
        low =  DailyData.getLow();
        change=DailyData.getChange();
        last=  DailyData.getLast();
        vol=   DailyData.getVolume();
        ihigh= (int) high;
        ilow= (int) low;
        ilast= (int) last;
        ichange= (int) change;
quoteModel.set(rday, ihigh, ilow, ilast, ichange, vol);
this.addQuote(quoteModel);
        }//for
}
```

The GetQuote methode implemtation in **QutoeSvrImpl.java**

```
public DataCollection GetQuote(int StockCode, int nrDays) throws
SQLException,RemoteException{

      StockData DailyRecord;
      DataCollection AllRecords= new DataCollection(StockCode,nrDays);
      Date RSDate=new Date();

            //Date RSDate;
      float RSHigh;
      float RSLow;
      float RSChange;
      float RSLast;
      int RSVol;

      Connection Con;
      Calendar Today=Calendar.getInstance();
      Calendar now=Calendar.getInstance();
      String StrDay;
      String StrNow;
      Date StartDay=new Date();
      Date DateToday=new Date();
```

```
      now=Today;
      DateToday=now.getTime();
      StrNow = java.text.DateFormat.getDateInstance().format(DateToday);

      Today.add(Calendar.DATE,-nrDays);

      StartDay=Today.getTime();
      StrDay = java.text.DateFormat.getDateInstance().format(StartDay);

      java.lang.String queryStmt="SELECT * FROM history WHERE
stock_code="+StockCode+" AND date>'"+StrDay+"' AND date<='"+StrNow+"'";
            try{
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");}
            catch(ClassNotFoundException e){System.out.println(e);
            };
            java.lang.String url="jdbc:odbc:sql_stock";
            Con=DriverManager.getConnection(url,"stock_client","");
            Statement stmt=Con.createStatement();
            ResultSet rs=stmt.executeQuery(queryStmt);
            System.out.println("Query executed ");
            while (rs.next()){

            java.sql.Date sqlDate=rs.getDate("date");
              RSDate=(java.util.Date) sqlDate;
              RSChange=rs.getFloat("change");
              RSHigh=rs.getFloat("high");
              RSLow=rs.getFloat("low");
              RSLast=rs.getFloat("last");
              RSVol=rs.getInt("volume");

              DailyRecord=new
StockData(RSDate,RSHigh,RSLow,RSLast,RSChange,RSVol);
                try {
                AllRecords.addQuote(DailyRecord);}
                catch (java.lang.Exception e){System.out.println(e);}

            }
            //clear
            rs.close();
            stmt.close();
            Con.close();
            System.out.println("send data successful");
            System.out.println("wait another call");
            return AllRecords;
      }
```

PART 5 Client Services

5.1 Overview

A series of personal information service was provided in this system. User able to check their stock and cash balance, transaction records through the Internet. The personal service will become more popular and important, that is a key of success for an Internet Application.

5.2 Transaction Statement

The transaction statement shows all the buy and sell transaction records. Each time the clients perform a success order, a transaction record will write to database. Here are the information will show in the transaction statement.

1. Transaction ID
2. Transaction Date
3. Stock Name
4. Buy Or Sell
5. Deal Price
6. Quantity
7. Total Price

5.3 Account Information

The account Information shows the cash balance and stock balance of the clients. In stock balance, beside the stock information will show; the real time stock price also will show. If the stock is gain money, it will show in blue color, otherwise it will show in red color for lost money. The following fields will shows:

1. Stock Code
2. Stock Name
3. Number of shares
4. The Buy in Price
5. The date of Buy in date
6. The real-time market price
7. The change compare by the market price and buy in price
8. The change percentage
9. The total gain/loss

When client try to buy a stock, before the transaction completed, the amount of money will held for the transaction. If the buy order was fail or rejected, that amount of money will release. Also this operation will done if client try to sell a stock, the amount of stock will held for the incomplete transaction.

5.4 WatchList

Watch List useful tools for stock investor. Client able to view several stocks real-time information at the same time and update the records by press refresh button. Client is able to add/remove their favorite stock.

Functions of Watch List:

1. Add a stock
2. Remove a stock
3. Refresh and display the latest stock information

5.5 Personal Information

Client is able to change their personal information, such as phone number, address by wed browser.

Functions of Personal Information

Personal information included:

1. Name
2. Address
3. Phone
4. Email address

1. View Personal Information

2. Change Personal Information

5.6 System Designs

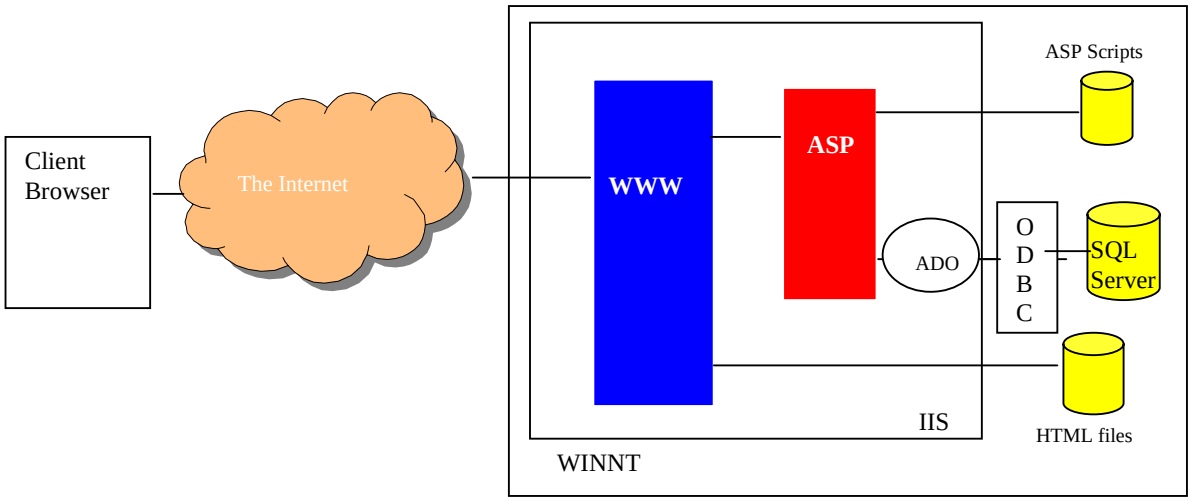
5.6.1 System Background

Language: ASP, VB Script, and JAVA Script

Platform: MS WINNT Server 4.0, IIS4.0

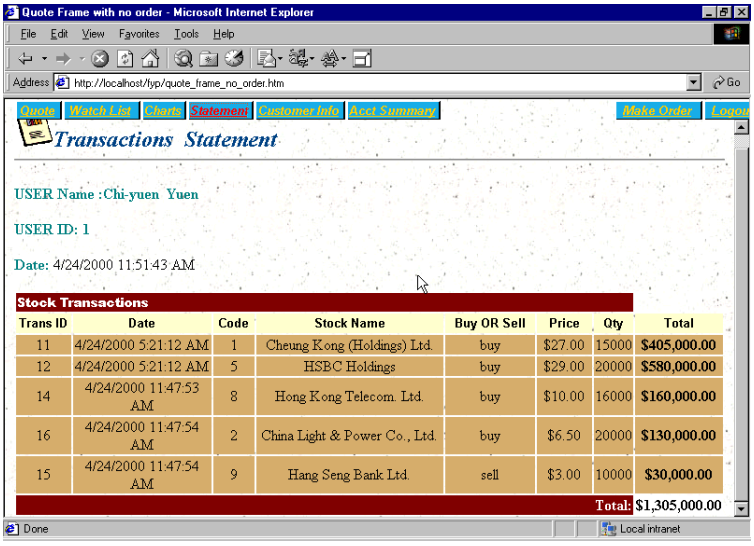
Database: MS SQL Server 6.5

5.6.2 System Architecture

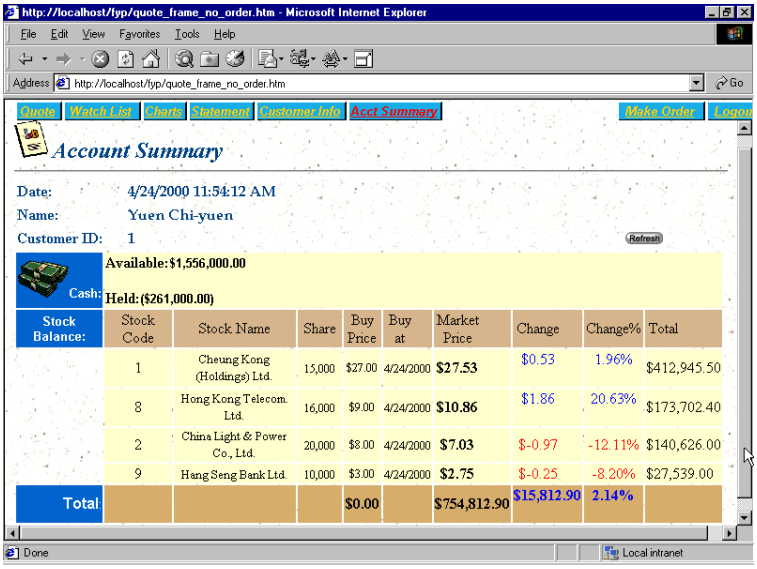


5.7 User Interface

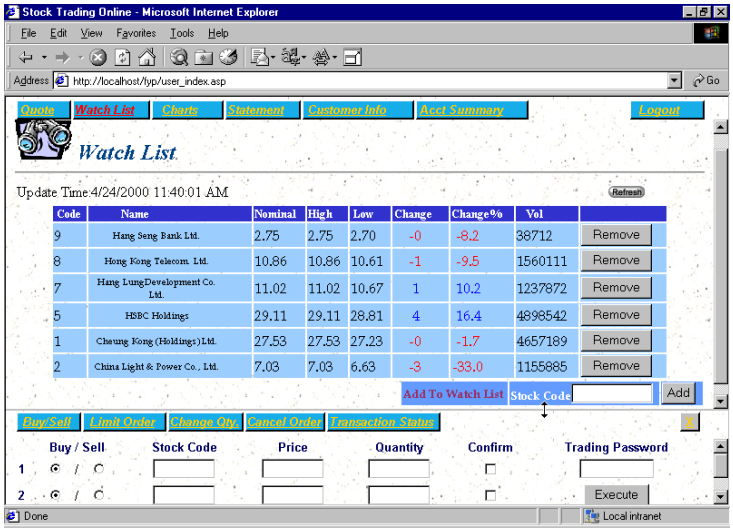
Transaction Statement



Account Information



WatchList



5.8 Selected Source Code

WatchList.asp

```
<%
NowTotal=0
OldTotal=0
CustomerID=Session("USERID")
strDbConnection="DATABASE=stock;DSN=sql_stock;UID=stock_client;Password=;”

'get the user watch list
SqlGetWatchList="SELECT * FROM watchlist where CustomerID="&CustomerID&""

'Response.Write(SqlGetStockBalance)
Set Con=Server.CreateObject("ADODB.Connection")
Con.Open strDbConnection
Set ListRS=Con.Execute(SqlGetWatchList)
%>
<html>
<head>
<title>Watch List</title>
</head>
</font><em><b><font color="#004080" size="5">Watch List</font></b></em>
<hr>
<TABLE border=0 cellPadding=1 cellSpacing=1 style="height: 27px"
width="718">
  <TR>
    <TD align="left" width="500">Update Time:<%Response.Write(NOW)%></TD>
    <TD align="left" width="102"><a href="watch_list.asp"><img border="0"
src="../img/refreshEng.gif" align="right"></a></TD></TR></TABLE>
<div align="center">
  <table border="0" width="680" height="42">
    <tr>
      <td width="2" align="middle" height="1" bgcolor="#3333CC">
        <p align="left"><FONT
      size=2 color="#FFFFFF"><b> Code</b> </FONT>  </p>
  </td>

      <td width="101" align="middle" height="1" bgcolor="#3333CC"><FONT
      size=2 color="#FFFFFF"><b>Name</b></FONT></td>
      ……
      ……
      <td width="53" align="middle" height="1" bgcolor="#3333CC"></td>
      </tr>
  <%
       For StockField =1 to 10
          Set StockID = ListRS.Fields(StockField)
          if StockID<>0 then
             'get real time Price
                  SqlGetRTData=             " SELECT r.*,c.CompanyName "
                  SqlGEtRTData=SqlGEtRTData+" FROM real_time_price AS r
,company AS c "
                  SqlGEtRTData=SqlGEtRTData+" WHERE
r.StockCode="&StockID&""
                  SqlGEtRTData=SqlGEtRTData+" AND r.StockCode=c.StockCode"
                  set PriceRS=Con.Execute(SqlGetRTData)
                  Nonimal=PriceRS.Fields("Price")
                  high=PriceRS.Fields("high")
                  low=PriceRS.Fields("low")
                  last=PriceRS.Fields("last_close")

                  change=Nonimal-last
                  changep=(change/last)*100
                  vol=PriceRS.Fields("turnover")

                  %>
```

```
      <tr>
        <form method="post"
action=remove.asp?removestockid=<%Response.write(StockField)%> >
        <td width="5" align="middle" height="2" bgcolor="#99CCFF"><font
face="Book Antiqua"><%Response.Write(StockId)%></font></td>
      ……
      ……
        <td width="38" align="middle" height="2" bgcolor="#99CCFF"><font
face="Book Antiqua"><%if (Change<0) Then %>
<p><font color="red"><%else %> <font
    color="blue"><%end if%>
<%Response.Write(FormatNumber(changep,1))%></font></td>
        <td width="40" align="middle" height="2" bgcolor="#99CCFF"><font
face="Book Antiqua"><%Response.Write(vol)%></font></td>
        <td width="40" align="middle" height="2" bgcolor="#99CCFF">
        <p><font face="Book Antiqua"><input type="submit" value="Remove"
name="butRemove"></font></p>
        </td>
      </tr>
      </FORM>
      <%
      END IF
      NEXT
      Con.Close()
      SET ListRS=Nothing
      SET PriceRS=Nothing%>

    </table>
<CENTER></CENTER>
</div>

<div align="right">
  <table border="0" width="336">
    <tr>
      <td width="113" bgcolor="#6699FF">
        <p align="left"><b><font color="#993366" size="2" face="Book
Antiqua">Add To Watch List</font></b></p>
      </td>
      <form method="POST" action="add.asp">
      <td width="160" bgcolor="#6699FF">
       <p align="left"><b><font color="#FFFFFF" size="2" face="Book
Antiqua">Stock Code<input type="text" name="AddStockID" size="11"></font>
       </b>
     </td>
      <td width="43" bgcolor="#6699FF">
        <p align="left"><font face="Book Antiqua"><input type="submit"
value="Add" name="butAdd"></font></p>
      </td>
    </tr>
    </form>
  </table>
</div>
</body>
</html>
```

add.asp for watchlist

```
<%
CustomerID=Session("USERID")
AddID=Request.Form("AddStockID")
strDbConnection="DATABASE=stock;DSN=sql_stock;UID=stock_client;Password=;"
SqlgetNumQuote="Select NumQuote FROM WatchList WHERE
CustomerID="&CustomerID&""
SqlGetBlankField="SELECT * FROM WatchList WHERE CustomerID="&CustomerID&""
Set Con=Server.CreateObject("ADODB.Connection")
Con.Open strDbConnection
SET NumRS=Con.Execute(SqlgetNumQuote)
NumQ=NumRS.Fields("numQuote")
IF NumQ>9 THEN
      Response.Write("You Already have Quote 9 Stocks")
ElSE
      NumQ=NumQ+1
      SET BlankFieldRS=Con.Execute(SqlGetBlankField)
      For StockField =1 to 10
             IF BlankFieldRS.Fields(StockField)=0 or
BlankFieldRS.Fields(StockField)=null THEN
                  Blank=StockField
             END IF
    NEXT
      SqlAddWL="UPDATE WatchList SET Stock"&Blank&"="&AddID&",
NumQuote="&NumQ&" WHERE CustomerID="&CustomerID&""
      'Response.Write(sqlAddWl)
      Con.Execute(SqlAddWL)
      Con.Close

      Response.Redirect("watch_list.asp")
END IF
%>
```

remove.asp in watchlist

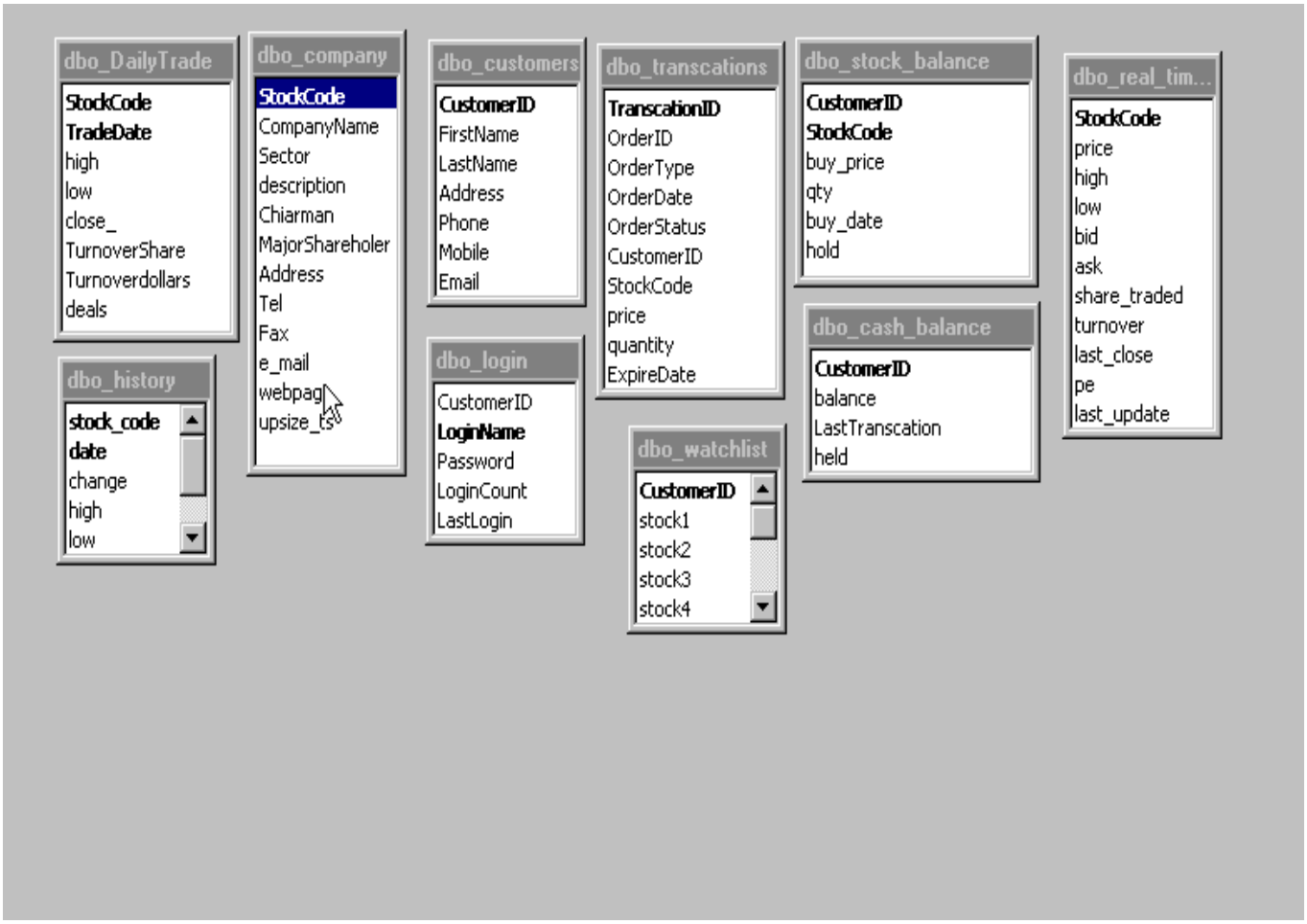
```
<%
CustomerID=Session("USERID")
RemoveID=Request.QueryString("RemoveStockID")

strDbConnection="DATABASE=stock;DSN=sql_stock;UID=stock_client;Password=;"

SqlgetNumQuote="Select NumQuote FROM WatchList Where
CustomerID="&CustomerID&""


SET Con=Server.CreateObject("ADODB.Connection")
Con.Open strDbConnection
SET NumRS=Con.Execute(SqlgetNumQuote)
NumQ=NumRS.Fields("numQuote")
NumQ=Numq-1
SqlRemoveWL="UPDATE WatchList SET Stock"&RemoveID&"=0 , numQuote="&NumQ&"
WHERE CustomerID="&CustomerID&""
Con.Execute(SqlRemoveWL)
Con.Close
Response.Redirect("watch_list.asp")
%>
```

Part 6 Database Design

In this Project, the MS SQL server 6.5 was employed. Nine tables were defined for different purpose. They defined as follow

Table Name	functions
Customers	Stored the customers' information
Login	Stored the customer's login password and login time
Company	Stored the list company information
Cash_balance	Stored the customer's cash balance and held cash
Stock_balance	Stored the customer's stock balance
Watch_list	Stored the customer's favorite stock for watchlist
Daily traded	Stored the daily traded recoded for all stock
Real_time_price	Stored the real time stock information and update by the random stock price generator
Transactions	Stored all the success transaction records

Each table's fields was defined as below



Part7 Further Development

7.1 Interface connection for AMS/3 OG

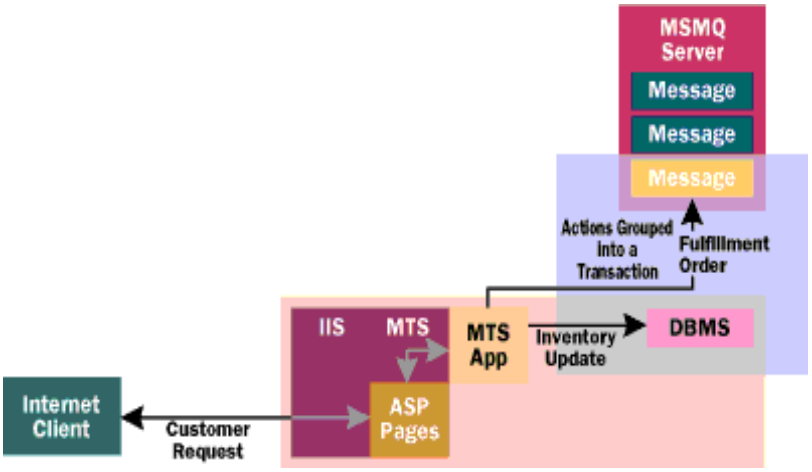
An interface program can be develop for translate the incoming message from OG and pack the order message as OG compatible message. By this step, this system can be easily merged to the coming AMS/3.

The detail specification of the AMS/3 OG ‘s can be find over the HKSE website.

7.2 Integrate with Microsoft Transaction Server

MTS is the agent between the components and application. It offers component functionality such as automatic transaction support, role base security, database connection etc. All the business logic can implement as a component and registry to the MTS. By ASP, we can create the server object to do the operation, such as buy and sell stock, user management. It not only provide a consistence business logic but also able to access the shared components in different scenery and computer language, such as Visual Basic.

MTS provides integration with MSMQ, a component can implement to check the new arrival message and do the validation automatically.

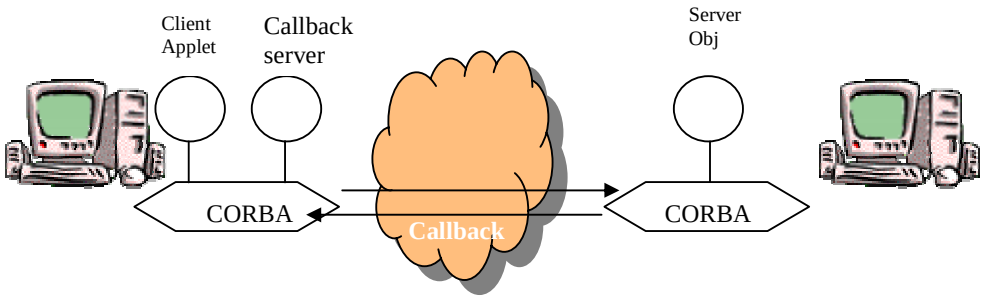


The structure of integration of IIS, MTS AND MSMQ

7.3 Real time push Stock Data by Callback

ASP implements the current design of the watch list, however user needed to press refreshes to update stock price. The new design is employ the **CORBA callback** and java applet to implement the WatchList. Each time when the stock price was updated, CORBA call back service will notify all login clients. By COBRA, the server can implement in C++ language

for better performance. In callback sceneries, there are not different between client and server, because server can invoke the method provide by client.



Conclusion

This project embedded the latest Internet Technology and middleware technology to construct a useable web enable system. It showed the operation of MSMQ and different Internet technology. In e-commerce world, reliable and complete IT Infrastructure is key to success. It is no doubt that middleware id one of the component, such as MSMQ, it provided a reliable and loosely coupled communication fashion. It is essential to build a reliable e-commerce application over the unreliable Internet.

Reference:

1.AMS/3 Overview by HKSE

2.Dickman, Alan (1998) Designing Applications with MSMQ, Addison Wesley

3.Francis, Fedorov, Harrison , Homer, Murphy, Sussman and Wood(1999) Professional Active Server Pages 2.0, Wrox.

4.Microsoft Message Queuing service White Paper: http://www.mircosoft.com/msmq

5.Martin Fowler , Kendall Scott (1998) UML Distilled Applying the Standard Object Modeling Language, Addison Wesley Press

6.Michael McKelvy (1995) Using Visual Basic 4 , Que Publish

7.Mktview Applet Interface,(1996) Softbear Inc

8.ORS Overview by HKSE

9.Serain, Daniel. (1999) Middleware,Springer

10.Thomas Connolly ,Carolyn Begg, Anne Strchan(1998)Database Systems , A Practical Approach to Design, Implementation and Management, Addison Wesley

11.Weber,Joseph (1999) Using Java2 Platform, Que Publish

12.Microsoft Message Queuing service Scenarios, Microsoft Corp.

13Transcation Server, A Guide to Reviewing MTS R2.0 ,.Microsoft Corp.

14.http://www.asp101.com

15.http://www.asptoday.com

16.http://www.Dynamicdrive.com

17.http://www.microsoft.com/msmq

18.http://www.boom.com

19.http://www.cash.com.hk

20.http://www.e-finet.com

21.http://www.shkonline.com

21.http://www.microsoft.com

22.http://www.messageq.com