

# **Commercial Mobile Device Service System**

## **Design Specification**

**Team member**                      Tao Ye, Yi Wang

**Advisor**                              Prof. Mahmood

**Date**                                      Dec 02, 2003

**File Name**                              Design\_Spec.doc

### **Revision History**

| Revision | Purpose of Changes   | Changed by with Date |
|----------|----------------------|----------------------|
| 1        | Initial Distribution | Tao Ye 12/02/2003    |
|          |                      |                      |
|          |                      |                      |

This master project is dedicated from the heart to my parents and Prof. Mahmood.

Appreciate the love and care from my parents and the help from Prof. Mahmood.

## **Content Page**

|                                                |    |
|------------------------------------------------|----|
| Introduction                                   | 4  |
| Design requirement.                            | 4  |
| Review of Enterprise Java Beans Architecture   | 5  |
| Client/Server Deployment Diagram of the System | 7  |
| System Architecture                            | 8  |
| Database Design                                | 10 |
| EJB Design                                     | 11 |
| Servlet Design                                 | 12 |
| Development Environment                        | 13 |
| Performance Discussion                         | 13 |
| Screen Snapshot                                | 15 |
| Reference                                      | 17 |
| Major Source Code                              | 18 |

## **Abstract**

Key words: EJB J2EE J2ME cellar weblogic

This article describes the theory and architecture of simulation for a commercial cell phone service system using J2EE and J2ME. The performance of the system is also discussed.

## **Introduction**

With the development of Java and wireless technology, more and more financial companies begin to utilize the wireless and real time system into the daily financial activities like stock transaction and banking service. And all these activities are closely related. So the need for a wireless device with comprehensive commercial functionality is coming up.

In this design solution, we try to design a commercial cell phone system, which involved 4 parties: Cell phone company, Stock company, bank, and cell phone. Except for normal cell phone functions, users also can use their cell phone as a stock trading terminals and banking service terminals.

In this project, the technologies involved include J2ME, Servlet, EJB, J2SE, SQL server, and development tools include Jbuilder and Weblogic.

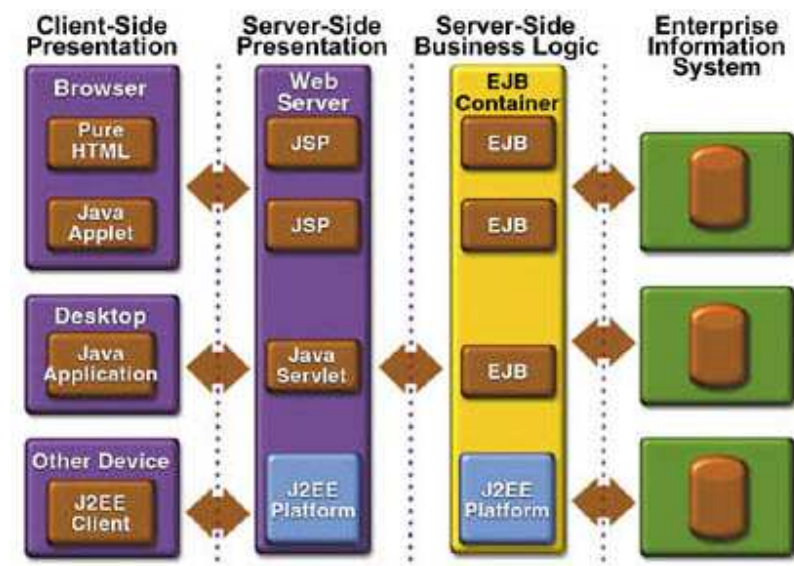
## **Design requirement.**

- **Stock Trading Service**  
Real time Stock price inquiry (Including max/min/current price), real time stock chart, trading index, market trade by current market price, limit trade by the price limit of users, trading state and account management. And daily transaction summary is also a plus.
- **Banking service**  
Account management (including saving, checking), balance inquiry, balance transfer.
- **Performance issue**  
EJB delivers the stability and scalability for business environment. To verify the performance of the system, a stress test needs to be done on server side. And brief discussion of the tuning the performance of weblogic is also a plus.

## **Review of Enterprise Java Beans Architecture**

Enterprise Java Beans (EJB) is a server-side component architecture that simplifies the process of building enterprise-class distributed component applications in Java. This component technology originally proposed by SUN Microsystem is agreed upon by the industry, supports portability and rapid development of server side applications. EJB components (enterprise beans) are deployed within application servers (EJB containers), which provide the needed middleware.

EJB 2.0 specification defines three different kinds of enterprise beans: session beans which model business processes, entity beans which are the object that caches database information, and message-driven beans which are similar to session beans but could be called only by sending messages to those beans. Enterprise beans are not full-fledged remote object but their invocations are intercepted by the EJB container and then delegated to the bean instances. At the interception point the following major services provided by the EJB container are also available: transaction management, persistence, resource management and component life cycle, security, concurrency, and remote accessibility.



EJB containers are responsible for managing enterprise beans and interact with beans by calling management methods as required. A multi-tier architecture scalability is enhanced when EJB container intelligently manages the needed resources across a variety of deployed components. These resources could be threads, socket connections, database connections, and more. It means that the EJB container is responsible for coordinating the entire effort of resource management as well as managing the deployment beans' life cycle. The interface between enterprise beans and the EJB container is a fundamental point of this technology which is described in detail by the EJB specification. Exact management schemes and their configuration attributes used by EJB container is

application server implementation specific. This makes evaluation of Application Servers offered by different vendors rather difficult and opens an area for interesting performance study.

EJB container management schemes supports persistency and database operations. Their activity depends on many attributes set in configuration of EJB Application Server. These attributes control JDBC operations or Container Managed Persistency (CMP) and Transaction Service performed by the Application Server and could substantially influence the overall system performance. The number of these attributes, and inconsistency of their semantic definition across different EJB Application Servers implementation, make the choice of setting which provides the best performance for the given application a nontrivial task.

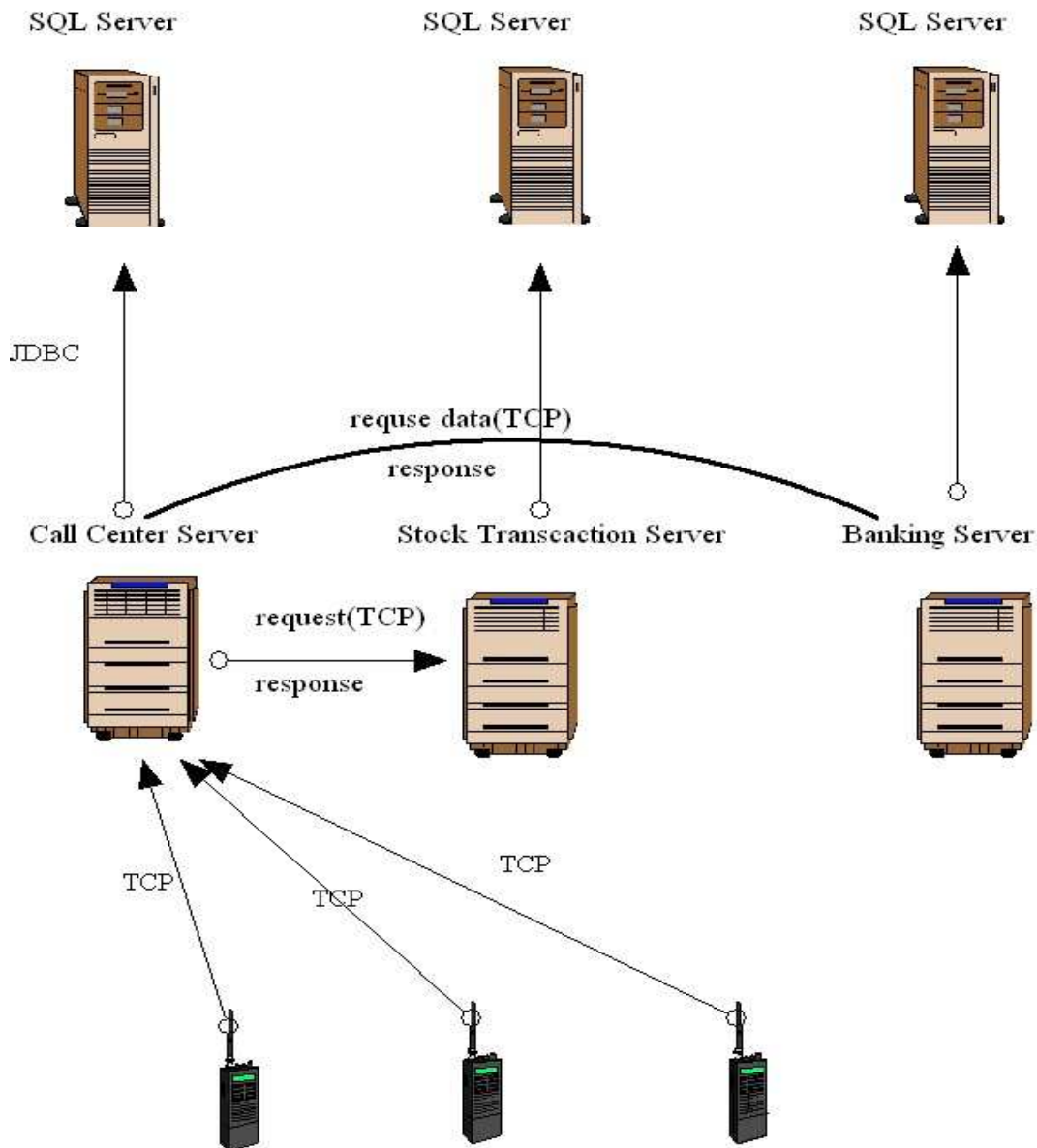
EJB based application performance is also very much depended on enterprise bean interfaces definition, their granularity, and data structure used for inter-communication. Thorough consideration of structuralization of the business processing and data access results in a proposal of design patterns. The design patterns have been actively exploited in the presented study to eliminate an additional overhead introduced by the inefficient usage of the EJB technology.

The *Session Facade* is one of the most popular EJB design patterns, which helps to obtain proper partition business logic and at the same time minimizes dependencies between a client and a server and forcing to execute business transaction in one networked call and in one transaction.

*Session Facade* pattern usage could results in reduction of remote calls. A pattern which addresses only data transfer reduction overhead is the *Value Object* pattern. *Value Object* encapsulates a set of attributes and provides *set/get* methods to access them. Value Objects are transported by value from the enterprise bean to the client component. When the client requests the enterprise bean for the business data this bean constructs the value object, populate it with the attribute values and pass it by value to the client. Client, who calls an enterprise bean, which uses a value object, makes only one remote call instead of numerous remote calls to get each attribute value in each call. The client receives *Value Object* and invokes locally *set/get* methods on this object for accessing attributes values. It is necessary to point out that the same pattern could be used to optimise access to data stored in database.

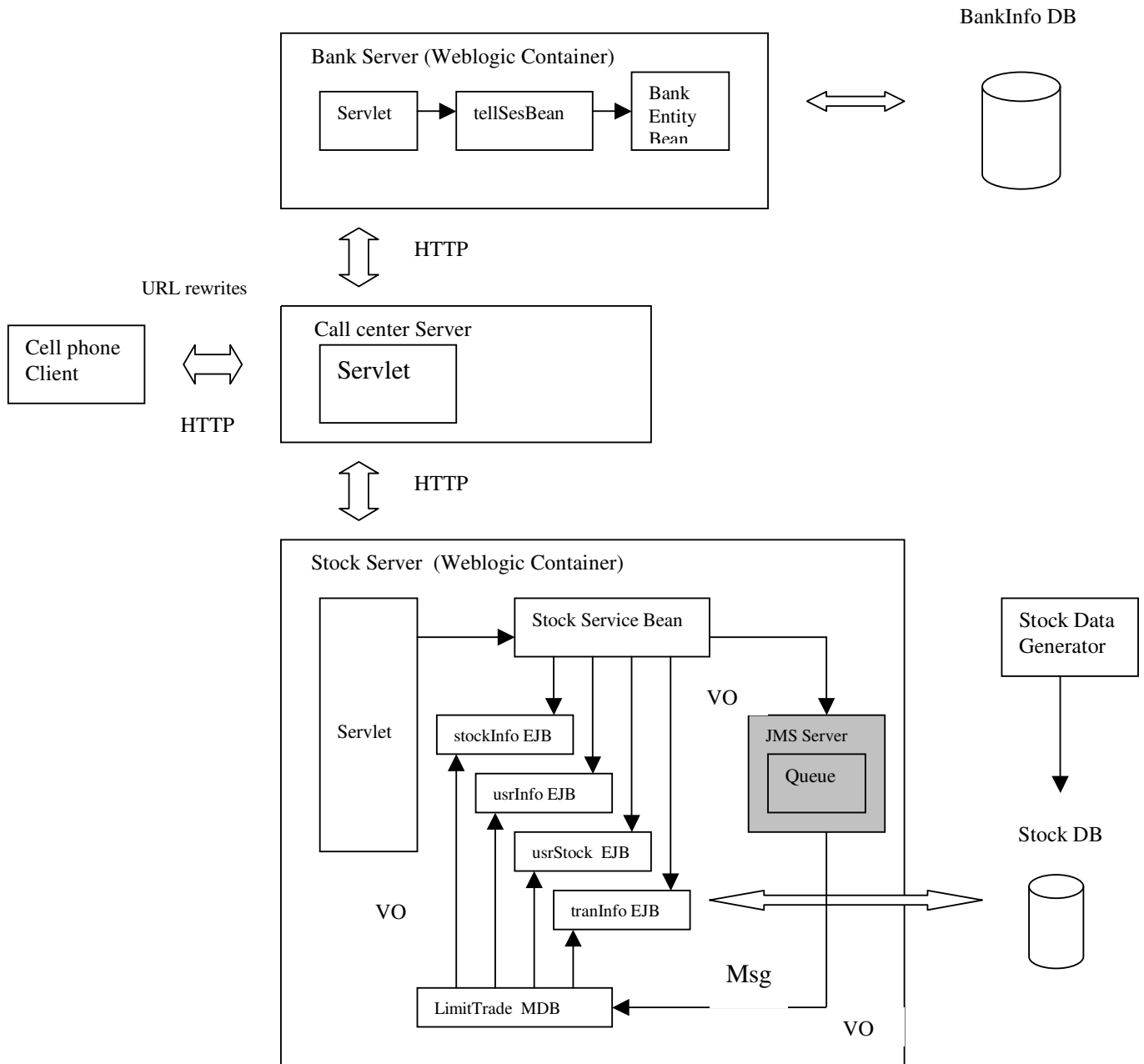
## Client/Server Deployment Diagram of the System

### Client/Server Deployment Diagram



From the diagram, we can see Call Center Server works not only as a server for cellular, but also as a client for the other two server.

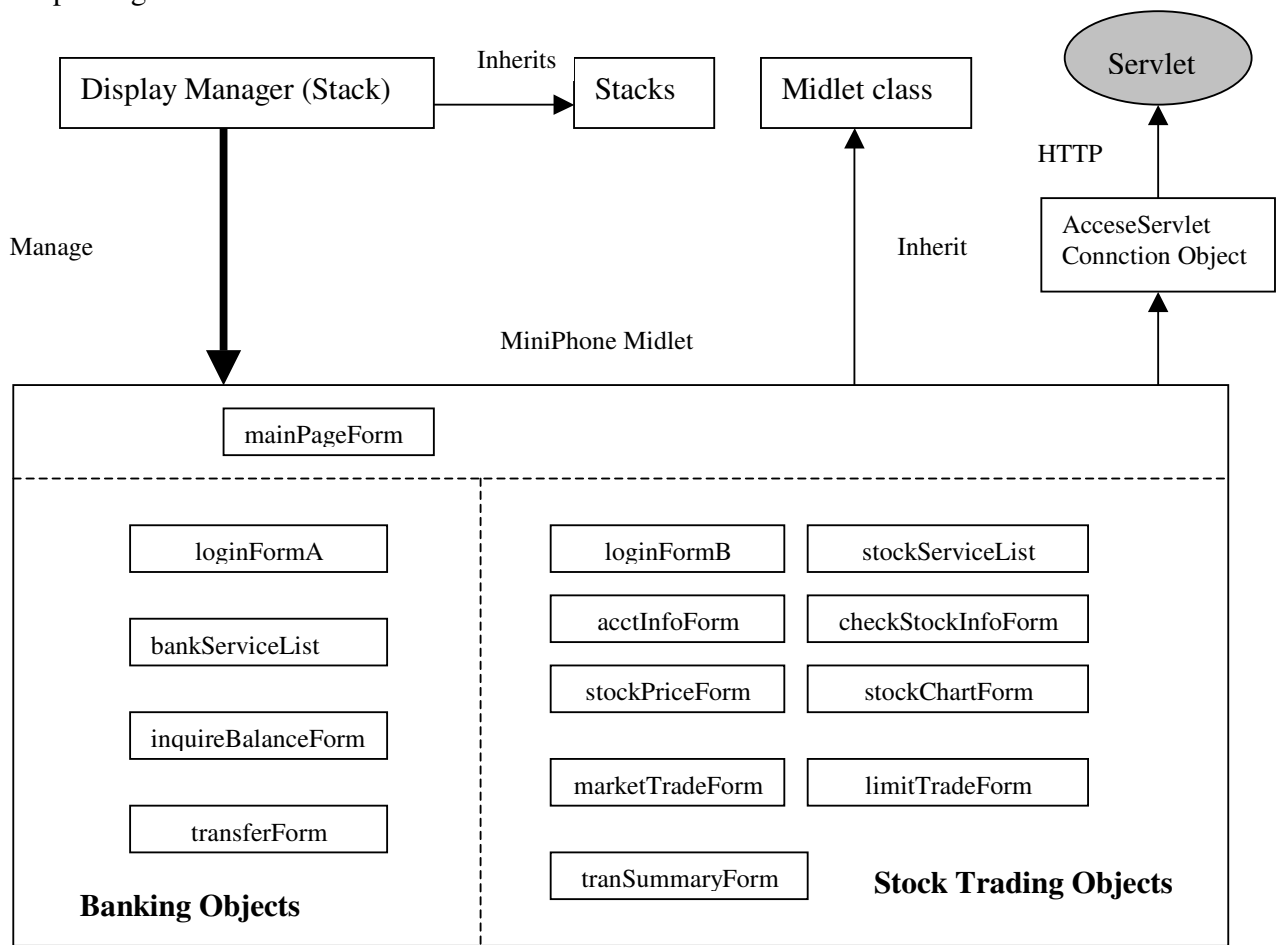
## System Architecture



Architecture of Server Side

From above diagram, we can see on the bank server, cellar client call the service method on servlet, and the servlet get the local interface of entity beans, by JINI service, and entity fulfill the transactions requirement by clients.

On stock server side, there are more entity beans and one more message driven bean. For inquire account, inquire stock information, market trade and transaction summary, these transaction are fulfilled through Stock service bean. We can see it serves as a *session facade*. For limit trade, once the limit is set by customer, due to the network latency, stock price and some other unpredictable reason, it might takes a long to for the stock price satisfied client's limit. So it is a good idea to put these transactions into a JMS queue. And use a message driven bean to fulfill the task of detecting stock price and updating the database.

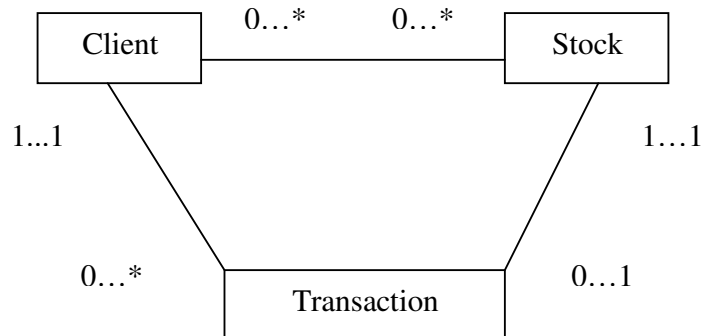


Architecture of client Side

From client's architecture diagram, we can see a display manager to control the business logic among these entire cellar screen page. This can let us easily go back and forth

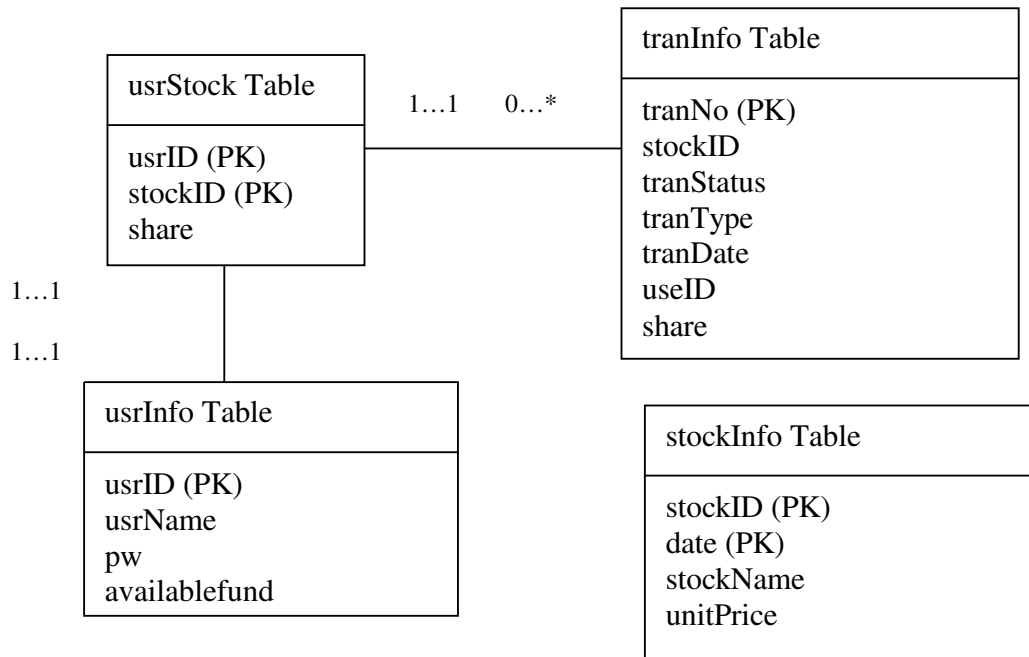
between different pages. For the stock chart function, cellars send the request data to servlet. After servlet get the return message from EJBs, it forwards the string to client in particular format. And then stockChartForm parses the string and generate the corresponding X,Y and draw the stock chart on canvas. It also support capabilities of zooming and flexible Y axis according to the highest and lowest price which give more clear stock charts.

### Database Design



ERD among the entities

There are 3 entities in the system. The relation among is show above. Client contains a lot of information. So it needs normalization. After that, the relation and properties of tables will like following:



## **EJB Design**

We are using stockService session bean as session façade. Its member functions are as following:

| stockService SLSB                                                                                                                                                         | bankService SLSB                            |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------|
| loginCheck()<br>InquireAcct()<br>InquireStockPrice()<br>getUsrStockChart()<br>marketTrade()<br>limitTrade()<br>tranSummary()<br>getAdjustedStockInfo()<br>limitTradeMDB() | identifyAcct()<br>getAcctVO()<br>transfer() |

1. Due to network latency and some unpredictable reason, when a client wants to get the stock price information through any method above, there could be a delay time. So how can we get the stock price if the time when we request is not equal to any time field in the stockInfo table? Here we use a getAdjustedStockInfo() method. It will do the followings:
  - a) Get a time range which is 30sec after current time
  - b) Get the collection of stock price whose time field is in above time range
  - c) Find the record whose time field is closest to current time.So even there is latency or the time of stock price record generated stock generator, we still can get the closest stock price data.
2. Method marketTrade() will do the following things:
  - a) Get the adjustedStockInfo.
  - b) Check transaction type
  - c) Get the available fund from usrInfo Table and check if user has enough fund to buy or if user has enough share of stock to sell.
  - d) If succeed, update following tables: usrStock, usrInfo, tranInfo.
  - e) All the transaction statement is in one try block, and we disable autoCommit() by weblogic, so if anyone of above statement fails, the statement won't be committed.
3. Method limitTrade () will just create the limitVO and send the request message to stockQueue; here stockService Session bean serves as a sender in the JMS service.

4. Method limitTradeMDB () will keep doing following in a while loop until it find the stock price satisfied user's limit.
  - a) Get the adjustedStockInfo.
  - b) Check user want to buy or sell a stock
  - c) Check if the stock price satisfies users' limit.
  - d) Check if the user have enough fund if he want to buy or if the user have enough share of stock if he wants to sell.
  - e) Update following tables if condition is satisfied: usrStock, usrInfo, tranInfo.
  - f) Wait for 10s if price does not match users' requirement.
  - g) All the transaction statement is atomic.

And entities beans' relation is also corresponding above relations.

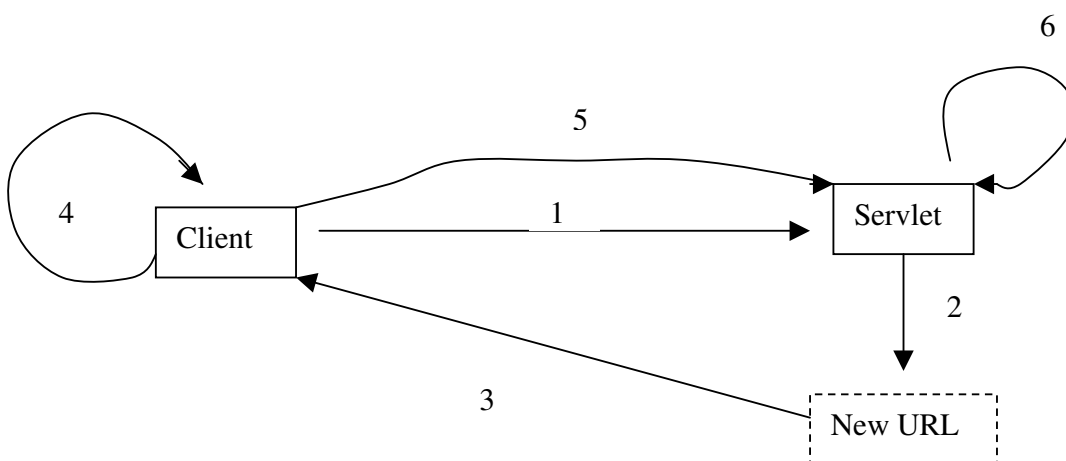
### **Servlet Design**

There are two servlet: miniPhoneServlet for banking and stockServlet for stock trading. The former is not complicate. Here we concentrate on the latter.

In the doGet() of stockServlet(), it does following:

- a) Determine if it is a new session.If it is, encoded the URL with customer ID
- b) Read the usrID if it is a old session
- c) Get the parameters passed in the Get Method
- d) Check request type
- e) Do the service request.

The procedure of URL rewriting is like following:



- 1) Client sends initial URL to servlet
- 2) Servlet encodes the URL if it is a new session
- 3) Servlet sets the new URL in http header and sends it back to client
- 4) Client update its URL variable
- 5) Client send message to servlet with new URL with session id.
- 6) Servlet get the session info through client http request with encoded URL

## **Development Environment**

In this development, Jbuilder and weblogic will be used because they handy at java programming and working well with each other.

Jbuilder need to be configured as following:

- 1) Change to memory size allocated to JVM, EJB deployment tool if you encounters java.lang.OutOfMemory error message.
- 2) Co the EJB container in Jbuilder
- 3) Configure the deployment tool
- 4) Configure the library for EJB and weblogic

Weblogic is high performance Web server and container. It needs to be configured as following:

- 1) Create connection with JDBC
- 2) Create TX Data Source; it support distributed transaction.
- 3) Create connection factory, JMS store, JMS server and destination for MDB.
- 4) More option can be changed in config.xml in the user domain directory.

More information can be retrieved at [www.bea.com](http://www.bea.com) and [www.borland.com](http://www.borland.com) .

## **Performance Discussion**

Weblogic is very popular due to it stability and stability in J2EE service. But sometime to get better performance, it needs to be tuned.

### **1. Time out value for JTA**

During our development, we have some problem of memory allocation which has already mentioned above, and we have also experienced following SQL exception frequently when try to set the price checking interval below 10s for limit trade.

```
java.sql.SQLException: The transaction is no longer active (status = Marked rollback. [Reason
=weblogic.transaction.internal.TimedOutException: Transaction timed out after 33 seconds
```

This happened because the default time out value of JTA in web logic is 30s. Unfortunately, default value equal to 30 seconds was to small for these tests and 300

seconds was chosen. After that, every 5s there is access to entity bean. Test result is 3 exceptions occurred for 174 calls to entity beans. And then 500s is chosen, it is much better. Last error was connected with J2EE application client, precisely with session beans stubs generated by Weblogic EJB compiler. These stubs have timeout (*RMI* timeout) value, which specifies the maximum time the client waits for response data from Weblogic server. The value by default is set to 30s seconds and if this is exceeded the exception is thrown and the client is unable to finish the business transaction. Unfortunately this error was the main drawback in these tests, especially when select data use case was performed.

## 2. Cache Size for Entity Beans

Weblogic 7.1 contains some extra features which enhance performance of CMP entity beans. This includes optimized loading (named groups and eager relationships fetching), db-is-shared (caching between transactions) options and read-only entities.

Unfortunately in case of Weblogic 7.1 certain errors occurred upon performance of some tests. Utilizing CMP 2.0 implementation method for 100 concurrent users may be given as example errors in selecting data use case. First error (Listing 1) was connected with EJB entity beans cache, which indicated that cache for entity beans was exceeded:

```
weblogic.ejb20.cache.CacheFullException
  at weblogic.ejb20.cache.EntityCache.put(EntityCache.java:363)
  at
weblogic.ejb20.manager.DBManager.getReadyBean(DBManager.java:312)
```

Listing 1 .Weblogic exception which indicates out of size for entities

Default Weblogic size cache for entity beans is set to 100 and when this number is greater then exception shown above, it is thrown by EJB container. To avoid this error appropriate cache size (max-beans-in-cache) for entity beans in weblogic-ejb-jar.xml deployment descriptor must be set. Simple formula to set this size is as follows  $\text{execute\_thread\_count} * \text{number\_of\_data\_returned}$ .

## 3. JDBC pool size

It there is high access volume, JDBC pool size also needs to be configured For this project, we test the EJB with 100 clients at the same the same time, there is no problem with the program. So we just choose the initial JDBC pool size

## **Screen Snapshot**



Bank Service Menu



Stock Service Menu



Account Info



Stock Chart



Zoomed In Stock Chart

Limit trade service

**Transaction Options**

☒ Trade Out  
☐ Trade In

**Stock ID:**  
GE

**Share:**  
2000

**Limit Price:**  
58

Back OK

Limit Trade

Transaction Summary

User ID: tom  
Stock ID: IBM  
Tran Status: Done  
Tran Type: buy  
Share: 19  
Date: 120203 153942

Stock ID: IBM  
Tran Status: Done  
Tran Type: sell  
Share: 19  
Date: 120203 153942

Back ↓

Transaction Summary

**Reference:**

1. John W. Muchow, *Core J2ME Technology and MIDP*, published by Sun press
2. Gail Anderson & paul Anderson, *Enterprise JavaBeans Component Architecture*. Published by Sun press.
3. Marty Hall, *Core Servlet and Java Server pages*, public by Sun press
4. Cay S. Horstmann & gary Cornell, *Core Java 2 technology (Volume I II)* , published by Sun press.
5. Zhao Jiang & Qiao xinliang, *J2EE Application Development with Weblogic & Jbuilder*, published by Fecit press
6. Harvey and Paul Deitel, *Java How to Program*, published by Prentice Hall
7. Rob Hawthorne, *SQL Server 2000 database Development from Scrach*, published by QUE press
8. Marcin Jarzab, Krzysztof Zieliński, *Database Access with EJB Application Servers Performance Study*.
9. Li Wei, *EJB Application Development Practice*.
10. Sudhansu Pati, *Borland® JBuilder® 7 and BEA WebLogic Server™ 7.0 Integration*
- 11 Jackwind Li Guojie, *Build your stock with J2ME*

## **Major Source Code**

### **A. Server Side**

#### **StockServiceHome.java**

```
package stockbean;

import javax.ejb.*;
import java.util.*;
import java.rmi.*;

public interface stockServiceHome extends javax.ejb.EJBHome {
    public stockService create() throws CreateException, RemoteException;
}
```

#### **stockService.java**

```
package stockbean;

import javax.ejb.*;
import java.util.*;
import java.rmi.*;
import java.math.*;

public interface stockService extends javax.ejb.EJBObject {
    public String loginCheck(String usrID, String pw) throws FinderException, RemoteException;
    public String inquireAcct(String usrID) throws FinderException, RemoteException;
    public String getUsrStockChart(String stockID, int day) throws FinderException, RemoteException;
    public String inquireStockPrice(String stockID) throws FinderException, RemoteException;
    public String marketTrade(String usrID, String stockID, int share, String tranType) throws FinderException, CreateException, RemoteException;
    public String limitTrade(String usrID, String stockID, int share, int limit, String tranType) throws RemoteException;
    public String tranSummary(String usrID, int day) throws FinderException, RemoteException;
}
```

#### **stockServiceBean.java**

```
package stockbean;

import javax.ejb.*;
import java.math.*;
import java.rmi.RemoteException;
import javax.naming.*;
import java.util.*;
```

```
import java.text.*;
import java.rmi.server.*;
import javax.jms.*;

public class stockServiceBean implements SessionBean {
    SessionContext sessionContext;
    private UserStockHome usrStock_Home;
    private UserInfoHome usrInfo_Home;
    private StockInfoHome stockInfo_Home;
    private TranInfoHome transaction_Home;
    private QueueConnection qConnect;
    private Queue stockQueue;
    private QueueSession qSession;
    private QueueSender qSender;
    public void ejbCreate() throws CreateException {
        /**@todo Complete this method*/
        try{
            Context initial=new InitialContext();
            System.out.println("Calling StockService Seesion Bean:ejbCreate()" );

            //Find LocalHome interface to usrStock EJB
            Object objref=initial.lookup("UserStock");
            usrStock_Home=(UserStockHome)objref;

            //Find LocalHome interface to usrInfo EJB
            objref=initial.lookup("UserInfo");
            usrInfo_Home=(UserInfoHome)objref;

            //Find LocalHome interface to stockInfo EJB
            objref=initial.lookup("StockInfo");
            stockInfo_Home=(StockInfoHome)objref;

            //Find LocalHome interface to usrStock EJB
            objref=initial.lookup("TranInfo");
            transaction_Home=(TranInfoHome)objref;

            QueueConnectionFactory qFactory=(QueueConnectionFactory)initial.lookup("stockJMSCF");
            qConnect=qFactory.createQueueConnection();
            stockQueue=(Queue)initial.lookup("stockQueue");
            qSession=qConnect.createQueueSession(false,Session.AUTO_ACKNOWLEDGE);
            qSender=qSession.createSender(stockQueue);

        }
        catch(Exception ex)
        {
            ex.printStackTrace();
            throw new EJBException(ex.getMessage());
        }
    }
}
```

```
    }
}
public void ejbRemove() {
    /**@todo Complete this method*/
}
public void ejbActivate() {
    /**@todo Complete this method*/
}
public void ejbPassivate() {
    /**@todo Complete this method*/
}
public void setSessionContext(SessionContext sessionContext) {
    this.sessionContext = sessionContext;
}
public String loginCheck(String usrID, String pw) throws FinderException {
    /**@todo Complete this method*/
    UserInfo user=usrInfo_Home.findByPrimaryKey(usrID);
    System.out.println("pw="+user.getPw().trim());
    if(user==null)
        return "Invalid";
    if(pw.equals(user.getPw().trim()))
        return "OK";
    else
        return "Invalid";
}
public String inquireAcct(String usrID) throws FinderException {
    /**@todo Complete this method*/
    Collection usrStock_c=usrStock_Home.findByUsrID(usrID);
    UserInfo usrInfo=usrInfo_Home.findByPrimaryKey(usrID);
    String      acct="User      ID:      "+usrID+"\nAvailabe      Fund:
"+usrInfo.getAvailableFund().intValue()+"\n";

    if(usrStock_c.size()>=1)
    {
        Iterator i=usrStock_c.iterator();
        UserStock userStock;
        while(i.hasNext())
        {
            userStock=(UserStock)i.next();
            acct=acct+"Stock ID: "+userStock.getStockID()+"\nShare: "+userStock.getShare()+"\n";
        }
        return acct;
    }
    else
        return "Invalid";
}
```

```
public String getUstrStockChart(String stockID, int day) throws FinderException {
    /**@todo Complete this method*/
    Collection c=stockInfo_Home.findByStockIDDay(stockID, new Integer(day));
    String s="";
    System.out.println("size="+c.size());
    if(c.size()>=1)
    {
        Iterator i=c.iterator();
        StockInfo stock;
        while(i.hasNext())
        {
            stock=(StockInfo)i.next();
            System.out.println("Time:"+stock.getStockTime());
            System.out.println("size="+c.size());
            s=s+stock.getUnitPrice().intValue()+"*"+stock.getStockTime().intValue()+"&";
        }
        s=s.substring(0,s.length()-1);
        return s;
    }
    else
        return "Invalid";
}

public String inquireStockPrice(String stockID) throws FinderException {
    /**@todo Complete this method*/
    StockInfo stock=getAdjustedStockInfo(stockID);
    if(!stock.equals(null))
    {
        String price="Stock ID: "+stockID+"\nStock Name: "+stock.getStockName()+"\nUnit Price:
"+stock.getUnitPrice()+"\nDate: "+stock.getStockDay().intValue()+"\nTime:
"+stock.getStockTime().intValue();
        System.out.println("price="+price);
        return price;
    }
    else
        return "Invalid";
}

public String marketTrade(String usrID, String stockID, int share, String tranType) throws
FinderException, CreateException {
    /**@todo Complete this method*/
    StockInfo stock=getAdjustedStockInfo(stockID);
    UserInfo userInfo=userInfo_Home.findByPrimaryKey(usrID);
    if(stock.equals(null))
        return "Invalid";
    UID tranNo=new UID();
    System.out.println("usrID="+usrID+"          stockID="+stockID+"          share="+share+"
tranType"+tranType);
```

```
Collection c=usrStock_Home.findByUsrIDStockID(usrID,stockID);
if(tranType.equals("buy"))
{
    if(userInfo.getAvailableFund().intValue()>=share*(stock.getUnitPrice().intValue()))
    {
        if(c.size()==1)
        {
            Iterator i=c.iterator();
            UserStock userStockLocal=(UserStock)i.next();
            userStockLocal.setShare(new BigDecimal(userInfo.getAvailableFund().intValue()-
            share*(stock.getUnitPrice().intValue())));
        }
        else
            usrStock_Home.create(usrID, stockID,new BigDecimal(share));

        transaction_Home.create(new BigDecimal(tranNo.hashCode()),stockID,new
        Integer(1),"buy", stock.getStockDay(),stock.getStockTime(),usrID,new BigDecimal(share));
        userInfo.setAvailableFund(new BigDecimal(userInfo.getAvailableFund().intValue()-
        share*(stock.getUnitPrice().intValue())));
        return "OK";
    }
    else
        return "NoFund";
}
else
{
    {
        if(c.size()!=1)
            return "NoStock";
        else
        {
            Iterator i=c.iterator();
            UserStock userStockLocal=(UserStock)i.next();
            if(userStockLocal.getShare().intValue()>=share)
            {
                System.out.println("I am in the sell case,new
share="+userStockLocal.getShare().intValue()-share));
                userStockLocal.setShare(new BigDecimal(userStockLocal.getShare().intValue()-share));
                transaction_Home.create(new BigDecimal(tranNo.hashCode()),stockID,new
                Integer(1),"sell", stock.getStockDay(),stock.getStockTime(),usrID,new BigDecimal(share));
                userInfo.setAvailableFund(new
                BigDecimal(userInfo.getAvailableFund().intValue()+share*(stock.getUnitPrice().intValue())));
                return "OK";
            }
            else
                return "OverShare";
        }
    }
}
}
```

```
public String limitTrade(String usrID, String stockID, int share, int limit, String tranType) {
    /**@todo Complete this method*/
    limitTradeVO limitVO=new limitTradeVO(usrID,stockID,share,limit,tranType);
    try{
        ObjectMessage msg=qSession.createObjectMessage();
        msg.setObject(limitVO);
        qSender.send(msg);
        System.out.println("Sending out Msg from StockService Session Bean.");
        return "OK";
    }
    catch(Exception ex)
    {
        ex.printStackTrace();
        return "Invalid";    //To be discussed
    }
}

public String tranSummary(String usrID, int day) throws FinderException {
    /**@todo Complete this method*/
    Collection c=transaction_Home.findByUsrIDDay(usrID,new Integer(day));
    String s="User ID: "+usrID;
    String tranStatus=null;
    if(c.size()>=1)
    {
        Iterator i=c.iterator();
        TranInfo tran;
        while(i.hasNext())
        {
            tran=(TranInfo)i.next();
            if(tran.getTranStatus().intValue()==1)
                tranStatus="Done";
            else
                tranStatus="Suspend";
            s=s+"\nStock ID "+tran.getStockID()+"\nTran Status: "+tranStatus+"\nTran Type: "+tran.getTranType()+"\nShare: "+tran.getShare()+"\nDate: "+tran.getTranDay()+"\n"+tran.getTranTime()+"\n";
        }
        System.out.println("summary="+s);
        return s;
    }
    else
        return "Invalid";
}
```

```
public stockbean.StockInfo getAdjustedStockInfo(String stockID) throws FinderException {
    /**@todo Complete this method*/
    Date date = new Date();
    String price=null;
    int minGap;
    Integer minTime_int,maxTime_int;
    StockInfo adjustedStock=null;
    Format formatter1= new SimpleDateFormat ("MMddyy");
    Format formatter2= new SimpleDateFormat ("HHmmss");
    Integer sysDay = new Integer(formatter1.format(date));
    int sysTime =(new Integer(formatter2.format(date))).intValue();
    //Following code is no use.

    int currentTime_sec=timeToSecond(date);
    minTime_int=new Integer(secondToTime(currentTime_sec-200000));
    maxTime_int=new Integer(secondToTime(currentTime_sec-240));

    Collection
    c=stockInfo_Home.findByStockIDDayTime(stockID,sysDay,minTime_int,maxTime_int);
    if(c.size()>=1)
    {
        Iterator i=c.iterator();
        StockInfo stock=(StockInfo)i.next();
        minGap=java.lang.Math.abs(sysTime-stock.getStockTime().intValue());
        while(i.hasNext())
        {
            stock=(StockInfo)i.next();
            if(java.lang.Math.abs(sysTime-stock.getStockTime().intValue())<minGap)
            {
                adjustedStock=stock;
                minGap=java.lang.Math.abs(sysTime-stock.getStockTime().intValue());
            }
        }
    }
    else
        System.out.println("Adjusted stockInfo not found");
    return adjustedStock;
}

public int secondToTime(int s) {
    /**@todo Complete this method*/
    int hour=(int)(s/3600);
    double min_original=(s/3600.0-hour) *60;
    int min=(int)min_original;
    int sec=(int)((min_original-min)*60);
    System.out.println("hour="+hour+" min="+min+" sec="+sec);
    return hour*10000+min*100+sec;
}
```

```
public int timeToSecond(Date date) {  
    /**@todo Complete this method*/  
    System.out.print("h="+date.getHours());  
    return date.getHours()*3600+date.getMinutes()*60+date.getSeconds();  
}  
  
}
```

#### stockTradeMDB.java

```
package stockbean;
```

```
import javax.ejb.*;  
import javax.jms.*;  
import javax.naming.*;  
import java.io.*;  
import java.rmi.*;  
import java.rmi.server.*;  
import javax.rmi.PortableRemoteObject;  
import java.util.*;  
import java.text.*;  
import java.math.BigDecimal;
```

```
public class marketTradeMDBBean implements MessageDrivenBean, MessageListener {  
    MessageDrivenContext messageDrivenContext;  
    UserStockHome userStock_Home;  
    TranInfoHome transaction_Home;  
    StockInfoHome stockInfo_Home;  
    UserInfoHome userInfo_Home;  
    public void ejbCreate() throws CreateException {  
        /**@todo Complete this method*/  
        try{  
            InitialContext jini=new InitialContext();  
            Object objref=jini.lookup("UserStock");  
            userStock_Home=(UserStockHome)objref;  
  
            objref=jini.lookup("TranInfo");  
            transaction_Home=(TranInfoHome)objref;  
  
            objref=jini.lookup("StockInfo");  
            stockInfo_Home=(StockInfoHome)objref;  
  
            objref=jini.lookup("UserInfo");  
            userInfo_Home=(UserInfoHome)objref;  
            System.out.println("I am in MDB's ejbCreate().");  
        }catch(Exception ex)  
        {  
            throw new EJBException(ex);  
        }  
    }  
}
```

```
}
}

public void ejbRemove() {
    /**@todo Complete this method*/
}

public void onMessage(Message msg) {
    /**@todo Complete this method*/
    // System.out.println("I am at the beginning of On Meesage()");
    StockInfo stockLocal;
    UserInfo userLocal;
    UserStock usrStockLocal;

    TranInfo initialTran;
    int availableFund;

    try{
        //Get the message from Queue
        ObjectMessage message=(ObjectMessage)msg;
        limitTradeVO limitVO=(limitTradeVO)message.getObject();

        //Get UserInfo Local Interface
        userLocal=userInfo_Home.findByPrimaryKey(limitVO.getUsrID());
        availableFund=userLocal.getAvailableFund().intValue();
        int stockPrice;
        Collection
        c=userStock_Home.findByUsrIDStockID(limitVO.getUsrID(),limitVO.getStockID());

        //Executing Limit trade in the loop until price limit is satisfied
        while(true)
        {
            stockLocal=getAdjustedStockInfo(limitVO.getStockID());

            stockPrice=stockLocal.getUnitPrice().intValue();

            //use UID to generate unique transaction No
            UID tranNo=new UID();
            //case of "buy" a stock
            if(limitVO.getTranType().equals("buy"))
            {
                System.out.println("Price="+stockPrice+" limit="+limitVO.getLimit());
                if(stockPrice<=limitVO.getLimit()) //limit is satisfied
                {
                    //check if fund is enough
```

```
if(availableFund>stockPrice*limitVO.getShare())
{
    if(c.size()==1)
    {
        Iterator i=c.iterator();System.out.println("Up to here");
        UserStock userStockLocal=(UserStock)i.next();
        userStockLocal.setShare(new
BigDecimal(userStockLocal.getShare().intValue()+limitVO.getShare()));
    }
    else
        userStock_Home.create(limitVO.getUsrID(),                limitVO.getStockID(),new
BigDecimal(limitVO.getShare()));

        userLocal.setAvailableFund(new BigDecimal(userLocal.getAvailableFund().intValue()-
limitVO.getShare()*(stockLocal.getUnitPrice().intValue())));
        transaction_Home.create(new
BigDecimal(tranNo.hashCode()),limitVO.getStockID(),new                Integer(1),"buy",
stockLocal.getStockDay(),stockLocal.getStockTime(),limitVO.getUsrID(),new
BigDecimal(limitVO.getShare()));
        System.out.println("Limit                trade:                buy
succeed.Time="+stockLocal.getStockTime().intValue()+" Price="+stockLocal.getUnitPrice());
        break;
    }
    else
    {
        transaction_Home.create(new
BigDecimal(tranNo.hashCode()),limitVO.getStockID(),new                Integer(0),"buy",
stockLocal.getStockDay(),stockLocal.getStockTime(),limitVO.getUsrID(),new
BigDecimal(limitVO.getShare()));
        throw new stockServiceException("NoFund.");
    }
}
//case of selling a stock
else
{
    System.out.println("Price="+stockPrice+" limit="+limitVO.getLimit());
    if(stockPrice>limitVO.getLimit())
    {
        //get user's original share of stock to be sold
        if(c.size()!=1)
            throw new stockServiceException("NoStock");
        else
        {
            System.out.println("I ma in sell case");
            Iterator i=c.iterator();
            UserStock userStockLocal=(UserStock)i.next();
            if(userStockLocal.getShare().intValue()>=limitVO.getShare())
            {
```

```

        userStockLocal.setShare(new        BigDecimal(userStockLocal.getShare().intValue()-
limitVO.getShare()));
        transaction_Home.create(new
BigDecimal(tranNo.hashCode()),limitVO.getStockID(),new        Integer(1),"sell",
stockLocal.getStockDay(),stockLocal.getStockTime(),limitVO.getUsrID(),new
BigDecimal(limitVO.getShare()));
        userLocal.setAvailableFund(new
BigDecimal(userLocal.getAvailableFund().intValue()+limitVO.getShare()*(stockLocal.getUnitPr
ice().intValue())));
        System.out.println("Limit        trade:        sell
succeed.Time="+stockLocal.getStockTime().intValue()+" Price="+stockLocal.getUnitPrice());
        break;
    }
    else
    {
        transaction_Home.create(new
BigDecimal(tranNo.hashCode()),limitVO.getStockID(),new        Integer(0),"sell",
stockLocal.getStockDay(),stockLocal.getStockTime(),limitVO.getUsrID(),new
BigDecimal(limitVO.getShare()));
        throw new stockServiceException("OverShare.");
    }
}
}
}
//isRightTime() keep checking if time interval two second is passed
isRightTime();
}
}
catch(Exception ex)
{
    System.out.println(ex.getMessage());
    ex.printStackTrace();
    throw new EJBException(ex);
}
}

```

```

public void setMessageDrivenContext(MessageDrivenContext messageDrivenContext) {
    this.messageDrivenContext = messageDrivenContext;
}

```

```

public stockbean.StockInfo getAdjustedStockInfo(String stockID) throws FinderException {
    /**@todo Complete this method*/
    Date date = new Date();
    String price=null;
    int minGap;
    Integer minTime_int,maxTime_int;

```

```
StockInfo adjustedStockLocal=null;
Format formatter1= new SimpleDateFormat ("MMddyy");
Format formatter2= new SimpleDateFormat ("HHmmss");
Integer sysDay = new Integer(formatter1.format(date));
int sysTime =(new Integer(formatter2.format(date))).intValue();

int currentTime_sec=timeToSecond(date);
minTime_int=new Integer(secondToTime(currentTime_sec-200000));
maxTime_int=new Integer(secondToTime(currentTime_sec-240));

//System.out.println("Before Calling Finder in getAdjustedStock()");
Collection
c=stockInfo_Home.findByStockIDDayTime(stockID,sysDay,minTime_int,maxTime_int);
//System.out.println("After Calling Finder in getAdjustedStock()");
if(c.size()>=1)
{
    Iterator i=c.iterator();
    StockInfo stock=(StockInfo)i.next();
    minGap=java.lang.Math.abs(sysTime-stock.getStockTime().intValue());
    while(i.hasNext())
    {
        stock=(StockInfo)i.next();
        if(java.lang.Math.abs(sysTime-stock.getStockTime().intValue())<minGap)
        {
            adjustedStockLocal=stock;
            minGap=java.lang.Math.abs(sysTime-stock.getStockTime().intValue());
        }
    }
}
else
    System.out.println("Adjusted stockInfo not found");
return adjustedStockLocal;
}

public int secondToTime(int s)
{
    /**@todo Complete this method*/
    int hour=(int)(s/3600);
    double min_original=(s/3600.0-hour) *60;
    int min=(int)min_original;
    int sec=(int)((min_original-min)*60);
    // System.out.println("hour="+hour+" min="+min+" sec="+sec);
    return hour*10000+min*100+sec;
}

public int timeToSecond(Date date)
{

```

```
    /**@todo Complete this method*/
    return date.getHours()*3600+date.getMinutes()*60+date.getSeconds();
}

public void isRightTime()
{
    Date beginTime=new Date();
    //get start time
    int
beginTimeInt=beginTime.getHours()*3600+beginTime.getMinutes()*60+beginTime.getSeconds(
);
    Date currentTime;
    int currentTimeInt;
    do{
        currentTime=new Date();
        //get current time

currentTimeInt=currentTime.getHours()*3600+currentTime.getMinutes()*60+currentTime.getSe
conds();
        //compare the current time to start time,if pass five seconds,finish
        if((currentTimeInt-beginTimeInt)>10)
            break;
    }while(true);
}
}
```

#### limitTradeVO.java

```
package stockbean;

import java.io.*;
/**
 * <p>Title: </p>
 * <p>Description: </p>
 * <p>Copyright: Copyright (c) 2003</p>
 * <p>Company: </p>
 * @author unascribed
 * @version 1.0
 */

public class limitTradeVO implements Serializable {
    private String stockID,tranType,usrID;
    private int share,limit;
    public limitTradeVO(String usrID,String stockID, int share, int limit, String tranType) {
        this.usrID=usrID;
        this.stockID=stockID;
        this.share=share;
    }
}
```

```
        this.limit=limit;
        this.tranType=tranType;
    }
    protected String getUsrID()
    {
        return this.usrID;
    }

    protected String getStockID()
    {
        return this.stockID;
    }
    protected int getShare()
    {
        return this.share;
    }

    protected int getLimit()
    {
        return this.limit;
    }

    protected String getTranType()
    {
        return this.tranType;
    }
}
```

#### stockServlet.java

```
package stockbean;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.net.*;
import javax.rmi.PortableRemoteObject;
import javax.naming.*;
import java.lang.*;
import java.math.*;

/**
 * <p>Title: </p>
 * <p>Description: </p>
 * <p>Copyright: Copyright (c) 2003</p>
 * <p>Company: </p>
```

```
* @author unascribed
* @version 1.0
*/
```

```
public class stockServlet extends HttpServlet {
    static final private String CONTENT_TYPE = "text/vnd.wap.wml";
    static final private String DOC_TYPE = "<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD
WML 1.2//EN\" \"http://www.wapforum.org/DTD/wml12.dtd\">";
    private stockServiceHome stockServiceHomeObject=null;
    private stockService stockServiceObject=null;

    //Initialize global variables
    public void init() throws ServletException {
        try
        {
            //Get home interface of bank session bean
            InitialContext ctx = new InitialContext();
            Object ref = ctx.lookup("stockService");
            stockServiceHomeObject = (stockServiceHome ) PortableRemoteObject.narrow(ref,
stockServiceHome .class);
            stockServiceObject=stockServiceHomeObject.create();
        }
        catch(Exception ex)
        {
            System.out.println("Initial session bean error.");
        }
    }
    //Process the HTTP Get request

    //Process the HTTP Post request
    protected void sendOut(HttpServletResponse res,String invalidFlag,String alert,String outStr)
throws ServletException, IOException
    {
        if(outStr.equals(invalidFlag))
        {
            res.sendError(res.SC_BAD_REQUEST,alert);
            return;
        }
        res.setContentType(CONTENT_TYPE);
        PrintWriter out=res.getWriter();
        out.print(outStr);
        out.close();
    }

    public void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType(CONTENT_TYPE);
    }
}
```

```
//Return msg from doPost()
String retMsg=null;
try
{
    //Get session info
    HttpSession session=request.getSession(true);

    //If a new session,rewrite the URL for client
    if(session.isNew())
    {
        //get url arrive here
        String incomingURL=HttpUtils.getRequestURL(request).toString();

        //Encode by adding sessitn ID to URL
        String URLWithID=response.encodeURL(incomingURL);

        //send back a header to client with rewritten URL
        response.setHeader("Customer-newURL",URLWithID);
    }

    //Get usrID from session for all other service except "log in"
    String usrID=(String)session.getAttribute("usrID");

    //process service request accodring client request type
    String reqType=request.getParameter("requestType");
    System.out.println("reqType="+reqType);
    System.out.println("boolean="+reqType.equals("marketTrade"));

    if(reqType.equals("login"))
    {
        String ID=request.getParameter("userName");
        String pw=request.getParameter("password");

        //create seesion var: usrID when clients log in
        session.setAttribute("usrID",ID);

        //To be modified
        retMsg=stockServiceObject.loginCheck(ID,pw);
        System.out.println("retMsg="+retMsg);
        sendOut(response,"Invalid","Unable to locate account.",retMsg);
    }
    else if(reqType.equals("acctInfo"))
    {
        System.out.println("I am here:usrID="+usrID);
        retMsg=stockServiceObject.inquireAcct(usrID);
        sendOut(response,"Invalid","Account info not available",retMsg);
    }
    else if(reqType.equals("stockChart"))
```

```
{
    String stockID=request.getParameter("stockID");
    int stockDate=(new Integer(request.getParameter("stockDate"))).intValue();
    retMsg=stockServiceObject.getUsrStockChart(stockID,stockDate);

    //retMsg="20*0930&60*1010&63*1020&125*1100&65*1130&50*1200&45*1230&30*1300&
    40*1310&35*1320&42*1330&99*1400&60*1410&63*1420&70*1430&65*1439&50*1450&4
    5*1500&30*1512&40*1519&35*1540&42*1550&48*1600";
    sendOut(response,"Invalid","Stock chart not available",retMsg);
}
else if(reqType.equals("stockPrice"))
{
    String stockID=request.getParameter("stockID");
    retMsg=stockServiceObject.inquireStockPrice(stockID);
    sendOut(response,"Invalid","Stock price not available",retMsg);
}
else if(reqType.equals("marketTrade"))
{
    String tradeType=request.getParameter("tradeType");
    String stockID=request.getParameter("stockID");
    int share=(new Integer(request.getParameter("share"))).intValue();
    System.out.println("tradeType="+tradeType+" stockID="+stockID+" share="+share);
    retMsg=stockServiceObject.marketTrade(usrID,stockID,share,tradeType);
    sendOut(response,"Invalid","Your transaction can not be processed ",retMsg);
}
else if(reqType.equals("limitTrade"))
{
    String tradeType=request.getParameter("tradeType");
    String stockID=request.getParameter("stockID");
    int share=(new Integer(request.getParameter("share"))).intValue();
    int limitPrice=(new Integer(request.getParameter("limit"))).intValue();
    retMsg=stockServiceObject.limitTrade(usrID,stockID,share,limitPrice,tradeType);
    sendOut(response,"Invalid","Your transaction can not be processed ",retMsg);
}
else if(reqType.equals("tranSummary"))
{
    int summaryDate=(new Integer(request.getParameter("date"))).intValue();
    retMsg=stockServiceObject.tranSummary(usrID,summaryDate);
    sendOut(response,"Invalid","Your transaction can not be processed ",retMsg);
}
}
catch(Exception e)
{
    System.out.println("Msg:"+e.toString());
}
}
```

```
public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    response.setContentType(CONTENT_TYPE);
    System.out.println("I am in doGet()");
    doPost(request,response);
}
//Clean up resources
public void destroy() {
}
}
```

**ejb-jar.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 2.0//EN"
"http://java.sun.com/dtd/ejb-jar_2_0.dtd">
<ejb-jar>
    <enterprise-beans>
        <session>
            <display-name>stockService</display-name>
            <ejb-name>stockService</ejb-name>
            <home>stockbean.stockServiceHome</home>
            <remote>stockbean.stockService</remote>
            <ejb-class>stockbean.stockServiceBean</ejb-class>
            <session-type>Stateless</session-type>
            <transaction-type>Container</transaction-type>
        </session>
        <entity>
            <display-name>StockInfo</display-name>
            <ejb-name>StockInfo</ejb-name>
            <local-home>stockbean.StockInfoHome</local-home>
            <local>stockbean.StockInfo</local>
            <ejb-class>stockbean.StockInfoBean</ejb-class>
            <persistence-type>Container</persistence-type>
            <prim-key-class>stockbean.StockInfoPK</prim-key-class>
            <reentrant>False</reentrant>
            <cmp-version>2.x</cmp-version>
            <abstract-schema-name>StockInfo</abstract-schema-name>
            <cmp-field>
                <field-name>stockID</field-name>
            </cmp-field>
            <cmp-field>
                <field-name>stockName</field-name>
            </cmp-field>
            <cmp-field>
                <field-name>unitPrice</field-name>
            </cmp-field>
            <cmp-field>
```

```
<field-name>stockDay</field-name>
</cmp-field>
<cmp-field>
  <field-name>stockTime</field-name>
</cmp-field>
<query>
  <query-method>
    <method-name>findByStockIDDay</method-name>
    <method-params>
      <method-param>java.lang.String</method-param>
      <method-param>java.lang.Integer</method-param>
    </method-params>
  </query-method>
  <ejb-ql>SELECT OBJECT (o) FROM StockInfo o WHERE o.stockID=?1 AND
o.stockDay=?2</ejb-ql>
</query>
<query>
  <query-method>
    <method-name>findByStockIDDayTime</method-name>
    <method-params>
      <method-param>java.lang.String</method-param>
      <method-param>java.lang.Integer</method-param>
      <method-param>java.lang.Integer</method-param>
      <method-param>java.lang.Integer</method-param>
    </method-params>
  </query-method>
  <ejb-ql>SELECT OBJECT (o) FROM StockInfo o WHERE o.stockID=?1 AND
o.stockDay=?2 AND o.stockTime BETWEEN ?3 AND ?4</ejb-ql>
</query>
</entity>
<entity>
  <display-name>UserInfo</display-name>
  <ejb-name>UserInfo</ejb-name>
  <local-home>stockbean.UserInfoHome</local-home>
  <local>stockbean.UserInfo</local>
  <ejb-class>stockbean.UserInfoBean</ejb-class>
  <persistence-type>Container</persistence-type>
  <prim-key-class>java.lang.String</prim-key-class>
  <reentrant>False</reentrant>
  <cmp-version>2.x</cmp-version>
  <abstract-schema-name>UserInfo</abstract-schema-name>
  <cmp-field>
    <field-name>userID</field-name>
  </cmp-field>
  <cmp-field>
    <field-name>userName</field-name>
  </cmp-field>
  <cmp-field>
```

```
<field-name>pw</field-name>
</cmp-field>
<cmp-field>
  <field-name>availableFund</field-name>
</cmp-field>
<primkey-field>userID</primkey-field>
</entity>
<entity>
  <display-name>UserStock</display-name>
  <ejb-name>UserStock</ejb-name>
  <local-home>stockbean.UserStockHome</local-home>
  <local>stockbean.UserStock</local>
  <ejb-class>stockbean.UserStockBean</ejb-class>
  <persistence-type>Container</persistence-type>
  <prim-key-class>stockbean.UserStockPK</prim-key-class>
  <reentrant>False</reentrant>
  <cmp-version>2.x</cmp-version>
  <abstract-schema-name>UserStock</abstract-schema-name>
  <cmp-field>
    <field-name>userID</field-name>
  </cmp-field>
  <cmp-field>
    <field-name>stockID</field-name>
  </cmp-field>
  <cmp-field>
    <field-name>share</field-name>
  </cmp-field>
  <query>
    <query-method>
      <method-name>findByUsrID</method-name>
      <method-params>
        <method-param>java.lang.String</method-param>
      </method-params>
    </query-method>
    <ejb-ql>SELECT OBJECT (o) FROM UserStock o WHERE o.userID=?1</ejb-ql>
  </query>
  <query>
    <query-method>
      <method-name>findByUsrIDStockID</method-name>
      <method-params>
        <method-param>java.lang.String</method-param>
        <method-param>java.lang.String</method-param>
      </method-params>
    </query-method>
    <ejb-ql>SELECT OBJECT (o) FROM UserStock o WHERE o.userID=?1 AND
o.stockID=?2</ejb-ql>
  </query>
</entity>
```

```
<entity>
  <display-name>TranInfo</display-name>
  <ejb-name>TranInfo</ejb-name>
  <local-home>stockbean.TranInfoHome</local-home>
  <local>stockbean.TranInfo</local>
  <ejb-class>stockbean.TranInfoBean</ejb-class>
  <persistence-type>Container</persistence-type>
  <prim-key-class>java.math.BigDecimal</prim-key-class>
  <reentrant>False</reentrant>
  <cmp-version>2.x</cmp-version>
  <abstract-schema-name>TranInfo</abstract-schema-name>
  <cmp-field>
    <field-name>tranNo</field-name>
  </cmp-field>
  <cmp-field>
    <field-name>stockID</field-name>
  </cmp-field>
  <cmp-field>
    <field-name>tranStatus</field-name>
  </cmp-field>
  <cmp-field>
    <field-name>tranType</field-name>
  </cmp-field>
  <cmp-field>
    <field-name>tranDay</field-name>
  </cmp-field>
  <cmp-field>
    <field-name>tranTime</field-name>
  </cmp-field>
  <cmp-field>
    <field-name>usrID</field-name>
  </cmp-field>
  <cmp-field>
    <field-name>share</field-name>
  </cmp-field>
  <primkey-field>tranNo</primkey-field>
  <query>
    <query-method>
      <method-name>findByUsrIDDay</method-name>
      <method-params>
        <method-param>java.lang.String</method-param>
        <method-param>java.lang.Integer</method-param>
      </method-params>
    </query-method>
    <ejb-ql>SELECT OBJECT (o) FROM TranInfo o WHERE o.usrID=?1 AND
o.tranDay=?2</ejb-ql>
  </query>
</entity>
```

```
<message-driven>
  <display-name>Enterprise1</display-name>
  <ejb-name>marketTradeMDB</ejb-name>
  <ejb-class>stockbean.marketTradeMDBBean</ejb-class>
  <transaction-type>Container</transaction-type>
  <message-driven-destination>
    <destination-type>javax.jms.Queue</destination-type>
  </message-driven-destination>
</message-driven>
</enterprise-beans>
<assembly-descriptor>
  <container-transaction>
    <method>
      <ejb-name>stockService</ejb-name>
      <method-name>*</method-name>
    </method>
    <trans-attribute>Required</trans-attribute>
  </container-transaction>
  <container-transaction>
    <method>
      <ejb-name>StockInfo</ejb-name>
      <method-name>*</method-name>
    </method>
    <trans-attribute>Required</trans-attribute>
  </container-transaction>
  <container-transaction>
    <method>
      <ejb-name>UserInfo</ejb-name>
      <method-name>*</method-name>
    </method>
    <trans-attribute>Required</trans-attribute>
  </container-transaction>
  <container-transaction>
    <method>
      <ejb-name>UserStock</ejb-name>
      <method-name>*</method-name>
    </method>
    <trans-attribute>Required</trans-attribute>
  </container-transaction>
  <container-transaction>
    <method>
      <ejb-name>marketTradeMDB</ejb-name>
      <method-name>*</method-name>
    </method>
    <trans-attribute>Required</trans-attribute>
  </container-transaction>
  <container-transaction>
    <method>
```

```
<ejb-name>TranInfo</ejb-name>
<method-name>*</method-name>
</method>
<trans-attribute>Required</trans-attribute>
</container-transaction>
</assembly-descriptor>
</ejb-jar>
```

### **tellerSesHome**

```
package bankbean;
```

```
import javax.ejb.*;
import java.util.*;
import java.rmi.*;
```

```
public interface tellerSesHome extends javax.ejb.EJBHome {
    public tellerSes create() throws CreateException, RemoteException;
}
```

### **tellerSes.java**

```
package bankbean;
```

```
import javax.ejb.*;
import java.util.*;
import java.rmi.*;
import java.math.*;
```

```
public interface tellerSes extends javax.ejb.EJBObject {
    public boolean identifyAcct(Integer acctNo, String pw) throws FinderException,
    RemoteException;
    public acctVO getAcctVO(Integer AcctNo) throws FinderException, RemoteException;
    public String transfer(Integer acct, boolean ifSavingToChecking, BigDecimal amount) throws
    FinderException, RemoteException;
}
```

### **tellerSesBean.java**

```
package bankbean;
```

```
import javax.ejb.*;
import java.math.*;
import javax.naming.*;
```

```
public class tellerSesBean implements SessionBean {
    private SessionContext sessionContext;
    private BankTableHome bankHome;
```

```
public void ejbRemove() {
    /**@todo Complete this method*/
}
public void ejbActivate() {
    /**@todo Complete this method*/
}
public void ejbPassivate() {
    /**@todo Complete this method*/
}
public void setSessionContext(SessionContext sessionContext) {
    this.sessionContext = sessionContext;
}
public boolean identifyAcct(Integer acctNo, String pw) throws FinderException {
    /**@todo Complete this method*/
    BankTable account=bankHome.findByPrimaryKey(acctNo);
    String s=account.getPassword().trim();
    if(pw.equals(s))
        return true;
    else
        return false;
}

public acctVO getAcctVO(Integer AcctNo) throws FinderException {
    //Get account VO from Account entity bean
    BankTable account=bankHome.findByPrimaryKey(AcctNo);
    acctVO a=new
acctVO(AcctNo,account.getName(),account.getPassword(),account.getCheckAccount(),account.g
etSaveAccount());
    return a;
}

public String transfer(Integer acct, boolean ifSavingToChecking, BigDecimal amount) throws
FinderException {
    /**@todo Complete this method*/
    BankTable account=bankHome.findByPrimaryKey(acct);
    BigDecimal checking=account.getCheckAccount();
    BigDecimal saving=account.getSaveAccount();

    //Case of transfer fund from saving to checking
    if(amount.doubleValue()>0 && ifSavingToChecking== true)
    {
        //If there is enough fund in saving account
        if(saving.compareTo(amount)>=0)
        {
            account.setCheckAccount(checking.add(amount));
            account.setSaveAccount(saving.subtract(amount));
            return "ok";
        }
    }
}
```

```
        else
            return "NoFund";
            //System.out.println("Not enough balance in saving account: "+ account.getAccountNo());
        }
        //Case of transfer fund from checking to saving
        else if(amount.doubleValue()>0 && ifSavingToChecking == false)
        {
            if(checking.compareTo(amount)>=0)
            {
                account.setCheckAccount(checking.subtract(amount));
                account.setSaveAccount(saving.add(amount));
                return "ok";
            }
            else
                return "NoFund";
            //System.out.println("Not enough balance in checking account: "+ account.getAccountNo());
        }
        else
            return "InvalidInput";
            //System.out.println("The amount you want to tansfer is not correct.");
    }
}

public void ejbCreate() throws CreateException {
    /**@todo Complete this method*/
    try {
        //Get the handle of Entity Bean
        Context context = new javax.naming.InitialContext();
        Object object = context.lookup("BankTable");
        bankHome = (BankTableHome)object;
    }
    catch (NamingException ex)
    {
        ex.printStackTrace();
    }
    catch (ClassCastException ex)
    {
        ex.printStackTrace();
    }
    catch (Exception ex)
    {
        ex.printStackTrace();
    }
}
```

**acctVO.java**

```
package bankbean;

import java.util.*;
import java.math.BigDecimal;
/**
 * <p>Title: </p>
 * <p>Description: </p>
 * <p>Copyright: Copyright (c) 2003</p>
 * <p>Company: </p>
 * @author unasccribed
 * @version 1.0
 */

public class acctVO implements java.io.Serializable{

    private Integer acctNo;
    private String name;
    private String acctPw;
    private BigDecimal checkingAcct;
    private BigDecimal savingAcct;

    public acctVO(Integer acctNo,String name,String acctPw,BigDecimal checkAcct,BigDecimal
savingAcct) {
        this.acctNo=acctNo;
        this.name=name;
        this.acctPw=acctPw;
        this.checkingAcct=checkAcct;
        this.savingAcct=savingAcct;

    }
    //getters
    public Integer getAcctNo() { return acctNo;}
    public String getName() { return name;}
    public String getAcctPw() { return acctPw; }
    public BigDecimal getCheckingAcct() { return checkingAcct; }
    public BigDecimal getSavingAcct() { return savingAcct; }

    //setter
    public void setAcctNo(Integer acctNo) { this.acctNo=acctNo; }
    public void setName(String name) { this.name=name; }
    public void setAcctPw(String acctPw) { this.acctPw=acctPw; }
    public void setCheckingAcct(BigDecimal checking) { this.checkingAcct=checking; }
    public void setSavingAcct(BigDecimal saving) { this.savingAcct=saving; }
}
```

## **B. Client Side**

### **DisplayManager.java**

```
/*DisplayManager.java*/
import javax.microedition.lcdui.*;
import java.util.*;

public class DisplayManager extends Stack
{
    private Display display;
    private Displayable mainDisplayable;

    public DisplayManager(Display display, Displayable mainDisplayable)
    {
        this.display=display;
        this.mainDisplayable=mainDisplayable;
    }
    public void pushDisplayable(Displayable newDisplayable)
    {
        push(display.getCurrent());
        display.setCurrent(newDisplayable);
    }
    public void home()
    {
        while(elementCount>1)
            pop();
        display.setCurrent(mainDisplayable);
    }
    public void popDisplayable()
    {
        display.setCurrent((Displayable)pop());
    }
    public void setAlertPage(Alert a)
    {
        display.setCurrent(a);
    }
}
```

### **MiniPhone.java**

```
/*MiniPhoneServlet.java*/

package miniphone;

import javax.servlet.*;
import javax.servlet.http.*;
```

```
import java.io.*;
import java.net.*;
import java.util.*;

/**
 * <p>Title: </p>
 * <p>Description: </p>
 * <p>Copyright: Copyright (c) 2003</p>
 * <p>Company: </p>
 * @author unascribed
 * @version 1.0
 */

public class miniPhoneServlet extends HttpServlet {
    static final private String CONTENT_TYPE = "text/vnd.wap.wml";
    static final private String DOC_TYPE = "<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD
WML 1.2//EN\" \"http://www.wapforum.org/DTD/wml12.dtd\">";
    //Initialize global variables
    public void init() throws ServletException {
    }
    //Process the HTTP Get request
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException {
        res.setContentType(CONTENT_TYPE);
        try
        {
            HttpSession session=req.getSession(true);

            String retMsg=null;
            String one=new String();
            String two=new String();
            String three=new String();
            String uName=new String();
            String uPassword=new String();
            one=req.getParameter("requestType");
            if(one.equals("login"))
            {
                two=req.getParameter("userName");
                session.setAttribute("userName",two);
                three=req.getParameter("password");
                session.setAttribute("userPassword",three);
                retMsg=checkUser(two,three);
            }
            else if(one.equals("transfer"))
            {
                two=req.getParameter("transferType");
```

```
        three=req.getParameter("amount");

        uName=(String)session.getAttribute("userName");
        uPassword=(String)session.getAttribute("userPassword");

        retMsg=balanceTransfer(two,three);
    }
    else if(one.equals("checkAccount"))
    {
        uName=(String)session.getAttribute("userName");
        uPassword=(String)session.getAttribute("userPassword");
        retMsg="5000";
    }
    else if(one.equals("saveAccount"))
    {
        uName=(String)session.getAttribute("userName");
        uPassword=(String)session.getAttribute("userPassword");
        retMsg="10000";
    }
    PrintWriter out=res.getWriter();
    out.print(retMsg);
    out.close();
}
catch(Exception e)
{
    System.err.println("Msg:" + e.toString());
}
}

private String checkUser(String name,String password)
{
    String ok;
    if(name.equals("wangyi") || password.equals("196374"))
        ok="ok";
    else
        ok="No such a user";
    return ok;
}
private String balanceTransfer(String type,String amount)
{
    String ok="ok";
    return ok;
}
//Clean up resources
public void destroy() {
}
}
```

**mainPage.java**

```
/*MainPage.java*/
import javax.microedition.lcdui.*;

public class MainPage extends List implements CommandListener
{
    private MiniPhone midlet;
    private Command cmExit;
    public MainPage(String title, int listType, MiniPhone midlet)
    {
        super(title,listType);
        this.midlet=midlet;
        append("Bank Service",null);
        append("Stock Service",null);
        cmExit=new Command("Exit",Command.EXIT,1);
        addCommand(cmExit);
        setCommandListener(this);
    }

    public void commandAction(Command c, Displayable s)
    {
        if(c==List.SELECT_COMMAND)
        {
            switch(getSelectedIndex())
            {
                case 0:
                    midlet.displayMgr.pushDisplayable(midlet.fmLoginA);
                    break;
                case 1:
                    midlet.displayMgr.pushDisplayable(midlet.fmLoginB);
                    break;
            }
        }
        else if(c==cmExit)
        {
            midlet.exitMIDlet();
        }
    }
}
```

**stockChart.java**

```
/****** stockChart.java
******/
import java.lang.*;
import javax.microedition.lcdui.*;
```

```

import javax.microedition.io.*;
import java.io.*;
import java.util.*;

public class stockChart extends Canvas implements CommandListener
{
    private MiniPhone midlet;
    private Command cmHome;
    private Command cmBack;
    private Vector vec1,vec2,vec3;
    private int maxX,maxY;
    private String zoomFactor;
    private int count=0,rightMove=0,maxPrice=0,minPrice=-1;
        private accessServlet acServlet;
        private int nasdaqStartTime=93000,nasdaqEndTime=160000;

    public stockChart(MiniPhone midlet)
    {
        this.midlet=midlet;
        acServlet=new accessServlet(midlet);
        cmHome=new Command("Home",Command.SCREEN,1);
        cmBack=new Command("back",Command.BACK,1);
        addCommand(cmHome);
        addCommand(cmBack);
        setCommandListener(this);
        maxX=getWidth()-4;
        maxY=getHeight()-4;
        System.out.println("maxY="+maxY);
        zoomFactor="1";
        vec1=new Vector();vec2=new Vector();vec3=new Vector();
    }

    /* stockString coming from servlet should be in the format of
    Price*HourMin&Price*HourMin,for example,
    * 43*131032&48*151556 represents price is $43 at 13:10:32 and price is $48 at 15:15:56.
    Note: Max X and Max Y
    * depend on specific MIDP device.
    */
    public void commandAction(Command c,Displayable s)
    {
        vec1.removeAllElements();
        vec2.removeAllElements();
        vec3.removeAllElements();
        midlet.fmCheckStockInfo.stockID="";
        if(c==cmBack)
        {

```

```

        midlet.displayMgr.popDisplayable();
    }
    else
    {
        midlet.displayMgr.home();
    }
}

public void getStockVector(String stockString)
{
    //CursorIndex marks the pos of "&",and start makes pos of start pos of each token in the
stockString.
    int cursorIndex=0,start=0;
    String strToken,strToken1;
    int x,y,seperatorIndex;
    Integer priceI, hourI, minI,secI;
    Vector price=new Vector(),hour=new Vector(),min=new
Vector(),sec=new Vector();

    while(true)
    {
        cursorIndex=stockString.indexOf('&',start);
        if(cursorIndex!=-1)
        {
            strToken=stockString.substring(start,cursorIndex);
            seperatorIndex=strToken.indexOf('*');
            priceI=Integer.valueOf(strToken.substring(0,seperatorIndex));
            strToken1=strToken.substring(seperatorIndex+1);
            if(strToken1.length()==4)
                strToken1="00"+strToken1;
            else if(strToken1.length()==5)
                strToken1="0"+strToken1;
            System.out.println("st1="+strToken1);
            hourI=Integer.valueOf(strToken1.substring(0,2));
            minI=Integer.valueOf(strToken1.substring(2,4));
            secI=Integer.valueOf(strToken1.substring(4));
            System.out.println("h="+hourI.intValue());
            price.addElement(priceI);
            hour.addElement(hourI);
            min.addElement(minI);
            sec.addElement(secI);
            if(maxPrice<priceI.intValue())
                maxPrice=priceI.intValue();

            if(minPrice== -1)
                minPrice=priceI.intValue();
            if(minPrice>priceI.intValue())

```

```
minPrice=priceI.intValue();

start=cursorIndex+1;
}
else
{
    strToken=stockString.substring(start);
    seperatorIndex=strToken.indexOf('*');

    priceI=Integer.valueOf(strToken.substring(0,seperatorIndex));

    strToken1=strToken.substring(seperatorIndex+1);
    if(strToken1.length()==4)
        strToken1="00"+strToken1;
    else if(strToken1.length()==5)
        strToken1="0"+strToken1;

    hourI=Integer.valueOf(strToken1.substring(0,2));
    minI=Integer.valueOf(strToken1.substring(2,4));
    secI=Integer.valueOf(strToken1.substring(4));

    price.addElement(priceI);
    hour.addElement(hourI);
    min.addElement(minI);
    sec.addElement(secI);
    if(maxPrice<priceI.intValue())
        maxPrice=priceI.intValue();

    if(minPrice>priceI.intValue())
        minPrice=priceI.intValue();
    break;
}

}

for(int i=0;i<price.size();i++)

vec1.addElement(getStockXY((Integer)hour.elementAt(i),(Integer)min.elementAt(i),(Integer)sec.elementAt(i),(Integer)price.elementAt(i),maxPrice));

//When cursor reached the end of the string.
if(vec1.isEmpty())
    System.out.println("Error:Vector size is zero");

}

public stockXY getStockXY( Integer hour,Integer min,Integer sec,Integer price,int maxPrice)
```

```
{
    /*Nasdaq stock market starts at 9:30am and ends at 4:00pm, timeGap is calculated in
       min.Assume the highest price of a stock won't be over maxPrice.
    */

    int timeGap=timeToSecond(hour.intValue()*10000+min.intValue()*100+sec.intValue())-
timeToSecond(nasdaqStartTime);

    int X=(int)(timeGap*maxX/workTime);
    int Y=(int)((price.intValue()*maxY/(maxPrice-minPrice)*maxY/maxPrice)*3/4)*maxPrice/
(maxPrice-minPrice);
    return new stockXY(X,Y);
}

protected void zoom(String zoomFactor)
{
    int x=-1,y,x1,y1,x2,y2; //(x2,y2) is the intersaction of stock curve and line x=maxX
    stockXY sXY,sXYNext=null,s;

    if(zoomFactor.equals("1.5"))
    {
        if(!vec2.isEmpty())
            vec2.removeAllElements();
        if(!vec3.isEmpty())
            vec3.removeAllElements();
        for(int i=0;i<vec1.size();i++)
        {
            sXY=(stockXY)vec1.elementAt(i);
            if(i<vec1.size()-1)

            sXYNext=(stockXY)vec1.elementAt(i+1);
            if(sXYNext!=null)
            {
                {
                    x2=sXYNext.getX();
                    y2=sXYNext.getY();
                    if(sXY.getX()<=(int)maxX*2/3)
                    {

                        if(sXYNext.getX()<(int)maxX*2/3)
                        {
                            {
                                x=(int)(sXY.getX()*3/2);
                                y=sXY.getY();
                                vec2.addElement(new stockXY(x,y));
                            }
                        }
                        else if(sXYNext.getX()>=(int)maxX*2/3)
                        {
                            {
                                x=(int)(sXY.getX()*3/2);
                                y=sXY.getY();
```

```

vec2.addElement(new stockXY(x,y));

x1=maxX;
y1=(y2-y)*x1/(x2*3/2-x)+(y*x2*3/2-x*y2)/(x2*3/2-x);
vec2.addElement(new stockXY(x1,y1));
vec3.addElement(new stockXY(2,y1));
    }
}
else
{
    x=(int)(sXY.getX()*3/2%maxX);
    y=sXY.getY();
    if(x!=0)
        vec3.addElement(new stockXY(x,y));
}
else

vec3.addElement(new stockXY(maxX,y));

                                }
                                else
                                {

        x=(int)(sXY.getX()*3/2%maxX);
        y=sXY.getY();

        if(sXY.getX()*3/2>maxX&& sXY.getX()*3/2%2!=0)

                                vec3.addElement(new
stockXY(x,y));

                                else

                                vec3.addElement(new
stockXY(maxX,y));

                                }

    }
}
else if(zoomFactor.equals("2"))
{
    if(!vec2.isEmpty())

                                vec2.removeAllElements();
                                if(!vec3.isEmpty())
                                    vec3.removeAllElements();

                                for(int i=0;i<vec1.size();i++)

        {
            sXY=(stockXY)vec1.elementAt(i);

                                if(i<vec1.size()-1)

```

```

sXYNext=(stockXY)vec1.elementAt(i+1);
    if(sXYNext!=null)
        {
            x2=sXYNext.getX();
            y2=sXYNext.getY();
            if(sXY.getX()*2<=maxX)
                {
                    if(sXYNext.getX()*2<maxX)
                        {
                            x=(int)(sXY.getX()*2);
                            y=sXY.getY();
                            vec2.addElement(new stockXY(x,y));
                        }
                    else
                        {
                            x=sXY.getX()*2;
                            y=sXY.getY();
                            vec2.addElement(new stockXY(x,y));

                            x1=maxX;
                            y1=(y2-
y)*x1/(x2*2-x)+(y*x2*2-x*y2)/(x2*2-x);

                            vec2.addElement(new stockXY(x1,y1));
                            vec3.addElement(new stockXY(2,y1));
                        }
                    }
                else
                {
                    x=(int)(sXY.getX()*2%maxX);
                    y=sXY.getY();
                    if(x!=0)
                        vec3.addElement(new stockXY(x,y));
                    else
                        vec3.addElement(new stockXY(maxX,y));
                }
            }
        else
        {
            x=(int)(sXY.getX()*2%maxX);
            y=sXY.getY();
        }
    }

```

```

        if(sXY.getX()*2>maxX&& sXY.getX()*2%2!=0)
            vec3.addElement(new
stockXY(x,y));
            else
            vec3.addElement(new
stockXY(maxX,y));
        }
    }
    sXYNext=null;
}

protected void paint(Graphics g)
{
    int x1=0,x2=0,y1=0,y2=0;
    int cursorIndex=0,start=0;
    String strToken;

    // Due to a bug in MIDP 1.0.3 we need to force a clear of the
display
    g.setColor(255, 255, 255);          // White pen
    g.fillRect(0, 0, getWidth(), getHeight());

    g.setStrokeStyle(Graphics.SOLID);
    g.setColor(51, 153, 0); // Deep green
    for(int i=0;i<2;i++)
    {
        g.drawRect(2, maxX/4-1, maxX-i, maxY*3/4-i);
        g.drawRect(2+i, i+maxX/4-1, maxX-1, maxY*3/4-1);
    }

    g.setColor(90,90,128); //Blue
    g.setFont(Font.getFont(Font.FACE_SYSTEM,
Font.STYLE_BOLD, Font.SIZE_MEDIUM));
    g.drawString("Stock ID: "+midlet.fmCheckStockInfo.stockID,
10, 5, Graphics.TOP | Graphics.LEFT);
    g.drawString("High: $" +maxPrice+" Low: $" +minPrice, 10, 20, Graphics.TOP |
Graphics.LEFT);

    //draw auxiliary dotted line
    g.setStrokeStyle(Graphics.DOTTED);
    g.setGrayScale(60);
    g.setColor(0,0,128); //Blue

```

```
        for(int i=1;i<6;i++)
        {
            g.drawLine(maxX*i/6,maxY/4,maxX*i/6,maxY);
g.drawLine(4,maxY/4+maxY*3/4*i/6,maxX,maxY/4+maxY*3/4*i/6);
        }

        g.setColor(255,0,0); //Blue

        stockXY s;

        if(zoomFactor.equals("1"))
        {
            /*      for(int j=0;j<vec1.size();j++)
            {
                s=(stockXY)vec1.elementAt(j);
                System.out.println("x="+s.getX()+"
Y="+s.getY());

            } */
            draw(vec1,g);
        }
        else if(rightMove==0&&!zoomFactor.equals("1"))
        {
            /*      for(int j=0;j<vec2.size();j++)
            {
                s=(stockXY)vec2.elementAt(j);
                System.out.println("x="+s.getX()+"
Y="+s.getY());

            } */
            draw(vec2,g);
        }
        else if(rightMove==1&&!zoomFactor.equals("1"))
        {
            /*      for(int j=0;j<vec3.size();j++)
            { System.out.println("j="+j);
                s=(stockXY)vec3.elementAt(j);
                System.out.println("x="+s.getX()+"
Y="+s.getY());

            } */
            draw(vec3,g);
        }
    }

    //draw stock chart
    protected void draw(Vector v,Graphics gg)
    {
        gg.translate(2,maxY); //translate coordinate
```

```

                                stockXY p1,p2;
                                gg.setStrokeStyle(Graphics.SOLID);
                                gg.setColor(255,0,0);
                                // System.out.println("I am in paint.");
                                for(int i=0;i<(v.size()-1);i++)
                                {
                                    p1=(stockXY)v.elementAt(i);
                                    p2=(stockXY)v.elementAt(i+1);
                                    // System.out.println("in
drawing:("+p1.getX()+","+p1.getY()+") to (" +p2.getX()+","+p2.getY()+")");
                                    gg.drawLine(p1.getX(),0-p1.getY(),p2.getX(),0-p2.getY());
                                    // System.out.println("in draw,i="+i);
                                }
                                }

public void keyPressed(int keyCode)
{
    switch(getGameAction(keyCode))
    {
        case UP:
        {
            if(count==0)
            {
                zoomFactor="1.5";

                zoom(zoomFactor);
                repaint();
                count++;
            }
            else if(count==1)
            {
                zoomFactor="2";
                zoom(zoomFactor);
                repaint();
                count++;
            }
        }
        break;
    }
    case DOWN:
    {
        if(count==2)
        {
            zoomFactor="1.5";
            zoom("1.5");
            repaint();
        }
    }
}
```

```
        count--;
    }
    else if(count==1)
    {
        zoomFactor="1";
        repaint();
        count--;
    }
    break;
}
case LEFT:
{
    if(rightMove==1)
    {
        rightMove=0;
        repaint();
    }
    break;
}
case RIGHT:
{
    if(rightMove==0)
    {
        rightMove=1;
        repaint();
    }
    break;
}
default:
    ;
    //      System.out.println("count="+count);
}
}
```

```
protected void prepareData(String url,String requestData)
{
    String retStr="";
    try
    {
        retStr=acServlet.process(midlet.url,requestData);
    }
    catch(Exception e)
    {
        System.err.println("Msg:"+e.toString());
    }
    //      System.out.println("retStr="+retStr);
    // Just for test
}
```

```

        if(retStr.equals("Invalid")||retStr==null)
            showAlert();
        else
        {
            getStockVector(retStr);
        }
    }

    protected void showAlert()
    {
        Alert alError=new Alert("Error Message","Stock Chart is not
available.",null,AlertType.ERROR);
        alError.setTimeout(Alert.FOREVER);
        midlet.displayMgr.setAlertPage(alError);
    }

    public int timeToSecond(int time) {
        /**@todo Complete this method*/
        int sec=time%100;
        int min=(int)(time/100)%100;
        int hour=(int)(time/10000);
        return hour*3600+min*60+sec;
    }

}

/***** stockCordinate.java *****/
*****
* This class represents data struc for each point in stock chart. Internally it convert the time to
* a specific value on X axis and price axis in stock chart.
*/

class stockXY
{
    private int X,Y;
    public stockXY(int x,int y)
    {
        X=x;
        Y=y;
    }

    protected int getX()

```

```

    {
        return X;
    }

    protected int getY()
    {
        return Y;
    }
}

```

**limitTrade.java**

```

/***** limitTrade.java *****/
import javax.microedition.lcdui.*;
import javax.microedition.io.*;
import java.io.*;

public class limitTrade extends Form implements CommandListener
{
    private TextField tfStockID;
    private TextField tfShare;
    private TextField tfLimit;
    private ChoiceGroup cgTradeType;
    private Command cmBack;
    private Command cmOK;
    private MiniPhone midlet;
    private String retStr,requestData;
    private int cgIndex;
    private accessServlet acServlet;

    public limitTrade(String title,MiniPhone midlet)
    {
        super(title);
        this.midlet=midlet;
        acServlet=new accessServlet(midlet);
        tfStockID=new TextField("Stock ID:", "",15,TextField.ANY);
        tfShare=new TextField("Share:", "",15,TextField.NUMERIC);
        tfLimit=new TextField("Limit Price:", "",15,TextField.NUMERIC);
        cgTradeType=new ChoiceGroup("Tracsaction Options",Choice.EXCLUSIVE);
        cmBack=new Command("Back",Command.BACK,1);
        cmOK=new Command("OK",Command.OK,1);
        retStr=null; requestData=null;

        int index=cgTradeType.append("Trade Out",null);
        cgTradeType.append("Trade In",null);
    }
}

```

```
        cgTradeType.setSelectedIndex(index,true);
        cgIndex=append(cgTradeType);

        append(tfStockID);
        append(tfShare);
        append(tfLimit);

        addCommand(cmBack);
        addCommand(cmOK);
        setCommandListener(this);
    }

    public void commandAction(Command c,Displayable s)
    {
        String tradeType;
        if(c==cmBack)
        {
            midlet.displayMgr.popDisplayable();
        }
        else if(c==cmOK)
        {
            if(tfStockID.getString().equals("")||tfShare.getString().equals("")||tfLimit.getString().equals("")||tfLimit.getString().equals("0"))
            {
                showAlert1();
            }
            else
            {
                if(cgTradeType.getString(cgTradeType.getSelectedIndex()).equals("Trade Out"))
                {
                    tradeType="sell";
                }
                else
                {
                    tradeType="buy";
                }

                requestData="requestType="+tradeType+"&limitTrade="+tradeType+"&stockID="+tfStockID.getString()+"&share="+tfShare.getString();

                requestData=requestData+"&tradeType="+tradeType+"&limit="+tfLimit.getString();
                try
                {
                    processTrade();
                }
                catch(Exception e)
                {
                    System.err.println("msg:" + e.toString());
                }
            }
        }
    }
}
```

```
}

protected void processTrade()
{
    try{
        retStr=acServlet.process(midlet.url,requestData);
    }
    catch(Exception e)
    {
        System.err.println("Limit trade failed.");
    }
    StringItem si=new StringItem("", "");
    if(retStr.equals("Invalid"))
        showAlert();
    else
    {
        showInfo();
        //removeCommand(cmOK);
        //append(si);
    }
}

private void showAlert()
{
    Alert alError=new Alert("Error Message", "Transaction
Failed.", null, AlertType.ERROR);
    alError.setTimeout(Alert.FOREVER);
    midlet.displayMgr.setAlertPage(alError);
}

private void showAlert1()
{
    Alert alError=new Alert("Error Message", "Input information can't be
empty", null, AlertType.ERROR);
    alError.setTimeout(Alert.FOREVER);
    midlet.displayMgr.setAlertPage(alError);
}

private void showInfo()
{
    Alert alInfo=null;
    try
    {
        Image im = Image.createImage("/check.png");
    }
}
```

```
        alInfo=new Alert("Information"," Your request is being processed.\n Status'll
be available after \n  sometime.",im,AlertType.CONFIRMATION);
        alInfo.setTimeout(Alert.FOREVER);
    }
    catch(Exception e)
    {
        System.out.println("Unable to read png image.");
    }
    midlet.displayMgr.setAlertPage(alInfo);
}

}
```