Study on Application of Advanced J2EE in Stocks Exchange

XIAOYAN YANG，WEIFENG YIN, JIFANG LI, DELIAO YU
Computer Science and Information Technology College,
Zhejiang Wanli University, Ningbo, China, 315100
http://www.zwu.edu.cn

Abstract: Along with Java technology progress, the superiority of J2EE 5.0 is gradually appearing. Furthermore the invest of stocks is increasingly hot, so the paper tries to design and implement the system of stocks invest supervision by recent EJB3.0 under Hibernate and Struts conditions based on newly J2EE 5.0 . The paper explores the implementation of EJB persistence of data .The system is tested by several times, it runs effectively. In the mealtime, the paper discusses on the new basic functionality of JDK5.0 and J2EE 5 SDK features

Key-Words: multi-tier, platform-independent, J2EE 5, EJB, stocks, Hibernate, Struts, JDK5.0

1 Introduction

Software evolution is an iterative and incremental process that encompasses the modification and alteration of software models at different levels of abstraction. These modifications are usually performed independently, but the objects to which they are applied to, are in most cases mutually dependent. Inconsistencies may be created if the effects of an alteration are not properly identified, recorded, and propagated in other dependent models. For large systems, it is possible that there is a considerable number of such model dependencies, for which manual extraction is not feasible.

In response to this challenging problem, building upon the success of its Java programming language, Sun Microsystems developed a component-based architecture for the development and management of multilayer, server-centric applications known as the Java 2 Platform, Enterprise Edition (J2EE). This technology extended the existing Java 2 Platform, Standard Edition (J2SE) to provide platform-independent portability to enterprise applications in addition to other features like modularity and reusability.

2. The New Functionality of J2SE 5.0

Today Java technology is everywhere—in large enterprise systems, desktops, hand-held devices, and smart cards. Consequently, Java technology is the platform of choice for developers all over the world. All this has happened in the short time since the technology was first introduced in 1995. While there have been updates and enhancements since the first version, release 5.0 of the core Java platform brings to the table more language-level updates and other enhancements than at any other time, through the incorporation of a number of Java Specification Requests.

2.1 Autoboxing

As any Java programmer knows, you cannot put an int (or other primitive value) into a collection. Collections can only hold object references, so you have to box primitive values into the appropriate wrapper class (which is Integer in the case of int). When you take the object out of the collection, you get the Integer that you put in; if you need an int, you must unbox the Integer using the intValue method. All of this boxing and unboxing is a pain, and clutters up your code.

Problem: (pre-J2SE 5.0)

－ Conversion between primitive types and wrapper types (and vice-versa)

－ You need manually convert a primitive type to a wrapper type before adding it to a collection

```
int i = 22;
List l = new LinkedList();
l.add(new Integer(i));
```

The autoboxing and unboxing feature automates the process, eliminating the pain and the clutter.

Solution: Let the compiler do it

```
Byte byteObj=22;//Autoboxing conversion
int i = byteObj // Unboxing conversion
ArrayList<Integer>al=new ArrayList<Integer>();
al.add(22); // Autoboxing conversion
```

The Autoboxing and Unboxing feature of J2SE 5.0 eliminates the drudgery of manual conversion between primitive types (such as int or long) and wrapper types (such as Integer or Long).

The Java code that does not use AutoBoxing/UnBoxing functionality yet below.

```
    public class AutoBoxLab{
  /*Suppose the internal variables are in Wrapper types*/
  Integer iObj;
  Float fObj;
  Long lObj;
  Double dObj;
  public AutoBoxLab() {

  }
  public static void main( String[] args ) {
     AutoBoxLab a = new AutoBoxLab();

     /*You have to create instances of Wrapper classes first*/
     /*before you save them into the internal variables. This is called boxing*/
     a.iObj = new Integer( 22 );
     a.fObj = new Float( 22.0 );
     a.lObj = new Long ( 22L );
     a.dObj = new Double( 22 );

     /*In the following code, you are unboxing in order to  get primitive types*/
     System.out.println( " int Value of iObj is: " + a.iObj.intValue() );
     System.out.println( " float Value of iObj is: " + a.fObj.floatValue() );
     System.out.println( " long Value of iObj is: " + a.lObj.longValue() );
     System.out.println( " double Value of iObj is: " + a.dObj.doubleValue() );

  }
}
```

Modify the code so that it uses AutoBoxing and UnBoxing as shown below. The code that is highlighted with bold font represents the changed code.

```
  public static void main( String[] args ){
     AutoBoxLab a = new AutoBoxLab();
     //a.iObj = new Integer( 22 );
     a.iObj = 22; //Using AutoBoxing
     // a.fObj = new Float( 22.0 );
     a.fObj = 22.0f ;  // Using AutoBoxing
     // a.lObj = new Long ( 22L );
     a.lObj = 22L;  // Using AutoBoxing
     // a.dObj = new Double( 22 );
     a.dObj = 22d;  //Using AutoBoxing
     /*System.out.println( " int Value of iObj is:"+a.iObj.intValue());*/
     System.out.println( " int Value of iObj is: "+a.iObj);  //Using UnBoxing
     /*System.out.println(" float Value of iObjis:"+a.fObj.floatValue());*/
     System.out.println( " float Value of iObj is: " + a.fObj );  //Using UnBoxing
     /*System.out.println( " long Value of iObj is:"+a.lObj.longValue());*/
     System.out.println( " long Value of iObj is: " + a.lObj );  // Using UnBoxing
     /*System.out.println(" double Value of iObj is:"+a.dObj.doubleValue());*/
     System.out.println( " double Value of iObj  is: " + a.dObj );  //Using UnBoxing
```

2.2 Type-safe Enumerations

Pre-J2SE 5.0, the standard way to represent an enumerated type was the "int Enum pattern" as following:

```
// int Enum Pattern - has severe problems!
public static final int SEASON_WINTER = 0;
public static final int SEASON_SPRING = 1;
public static final int SEASON_SUMMER = 2;
public static final int SEASON_FALL  = 3;
```

This pattern has several problems as following:
Not typesafe - Since a season is just an int type, you can pass in any other int value where a season is required, or add two seasons together (which makes no sense).

No namespace - You must prefix constants of an "int enum" with a string (in this case SEASON_) to avoid collisions with other int enum types.

Brittleness - Because int enums are compile-time constants, they are compiled into an application that use them. If a new constant is added between two existing constants or the order is changed, the application must be recompiled. If it is not, it will still run, but its behavior will be undefined.

Printed values are uninformative - Because they are just ints, if you print one out, all you get is a number, which tells you nothing about what it represents, or even what type it is.

It is possible to get around these problems by using the Typesafe Enum pattern, but this pattern has its own problems: It is quite verbose, hence error prone, and its enum constants cannot be used in switch statements.

Type-safe Enumeration (enum) allows you to create enumerated types with arbitrary methods and fields. It provides all the benefits of the Typesafe Enum pattern without the verbosity and the error-proneness.

2.3 Generics

This long-awaited enhancement to the type system provides compile-time type safety check. For example, in pre-J2SE 5.0, when you take an element out of a Collection, you must cast it to the type of element that is stored in the collection. Besides being inconvenient, this is unsafe because the compiler cannot check whether the type you are casting to is a correct one or not. This means the casting can fail at runtime. In general, runtime failure is order of magnitude more expensive than compile time error.

Generics provides a way for you to communicate the element type of a collection (or simply type of a collection) to the compiler, so that it can be checked at compile time. Once the compiler knows the element type of the collection, the compiler can check whether you are inserting an element of correct type to the collection. When an element is taken out of a collection, the compiler insert the correct casts.

2.4 Enhanced for Loop

Problem: (pre-J2SE 5.0)

- Iterating over collections is tricky
- Often, iterator only used to get an element
- Iterator is error prone

(Can occur three times in a for loop)

Solution: Let the compiler do it

– New for loop syntax

for (variable : collection)

– Works for Collections and arrays

Enhanced for Loop Example below.

Old code:

```
void cancelAll(Collection c) {
for (Iterator i = c.iterator(); i.hasNext(); ){
TimerTask task = (TimerTask)i.next();
task.cancel();
}
}
```

New Code:

```
void cancelAll(Collection<TimerTask> c) {
for (TimerTask task : c)
task.cancel();
  }
```

2.5 Static Imports

Problem: (pre-J2SE 5.0)

– Having to fully qualify every static referenced from
external classes

Solution: New import syntax

– import static TypeName.Identifier;

– import static Typename.*;

– Also works for static methods and enums.

2.6 Variable Arguments

This Variable Arguments feature of J2SE 5.0 eliminates the need for manually boxing up argument lists into an array when invoking methods that accept variable-length argument lists.

In past releases, a method that took an arbitrary number of values required you to create an array and put the values into the array prior to invoking the method. For example, here is how one used the MessageFormat class to format a message:

```
Object[]arguments={
new Integer(7), new Date(),
"a disturbance in the Force"};

String result = MessageFormat.format(
"At {1,time} on {1,date}, there was {2} on
planet"+"{0,number,integer}", arguments);
// arguments is an array
```

It is still true that multiple arguments must be passed in an array, but the varargs feature automates and hides the process. Furthermore, it is upward compatible with preexisting APIs. So, for example, the MessageFormat.format method now has this declaration:

```
public static String format(String pattern,
        Object... arguments);
```

The three periods after the final parameter's type indicate that the final argument may be passed as an array or as a sequence of arguments. Varargs can be used only in the final argument position. Given the new varargs declaration for MessageFormat.format, the above invocation may be replaced by the following shorter and sweeter invocation:

```
String result = MessageFormat.format(
"At {1,time} on {1,date}, there was {2} on
planet " + "{0,number,integer}.",
7, new Date(), "a disturbance in the Force");
```

e.g **Math.sin(x)** becomes **sin(x).**

3. J2EE Framework

The goal of J2EE is to create a reliable, high-performance, secure, transactional, distributed architecture that would support concurrent user access. The architecture was also to allow the creation of applications that would seamlessly integrate with existing (presumably non-Java-based) systems while supporting future modifications and scaling to match demand with relative ease.

Advanced J2EE Platform Development explores the characteristics of real-world multi-tier software as implemented in J2EE, delivering a warts-and-all picture of the development environment as it's used for tying together user interfaces, business logic, databases, and legacy systems such as mainframes. This is a useful, eminently practical guide to J2EE software design.

The traditional Application of three-layer distributed framework is shown in Fig 1.

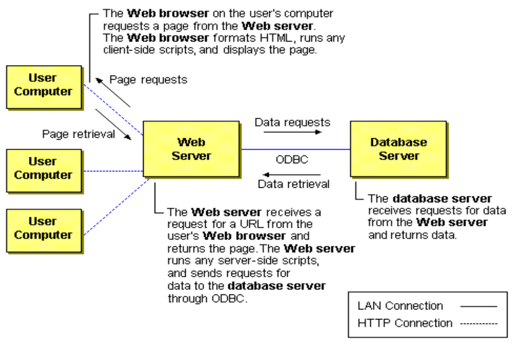

The Web browser on the user's computer requests a page from the Web server. The Web browser formats HTML, runs any client-side scripts, and displays the page.
User Computer
Page requests
Page retrieval
User Computer
User Computer
Web Server
Data requests
ODBC
Data retrieval
Database Server
The Web server receives a request for a URL from the user's Web browser and returns the page. The Web server runs any server-side scripts, and sends requests for data to the database server through ODBC.
The database server receives requests for data from the Web server and returns data.
LAN Connection
HTTP Connection

Fig. 1 Traditional three-layer framework

The figure 1 shows that computing is concentrated on the server under the environment of Browser-based and Internetworking.

Initially released in 1999, the specification has been revised several times, with the most recent release being the J2EE 1.5 specification.

The J2EE provides a design of framework that simplify the design of middle-ware of enterprise application, main including reusable componet,JSP,EJB ,simpler JDBC ,XML and so on. EJB (Enterprise JavaBeans) is the foundation of J2EE. As in the following Fig2.

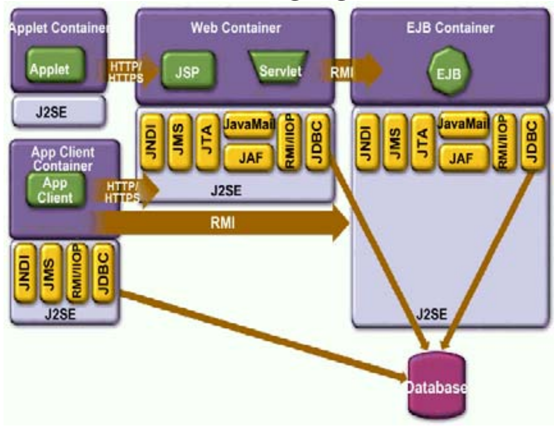


Fig. 2 J2EE multi-tier infrastructure

4. Java EE 5 SDK Features

Java Platform, Enterprise Edition 5 (Java EE 5) focuses on making development easier, yet retains the richness of the J2EE 1.4 platform. Offering new and updated features such as Enterprise JavaBeans (EJB) Technology 3.0, JavaServer Faces (JSF) Technology, and the latest web services APIs, Java EE 5 makes coding simpler and more straightforward, but maintains the power that has established Java EE as the premier platform for web services and enterprise application development.

The Java EE 5 SDK and Java Application Platform SDK provide support for Java EE 5 specifications, and the Java Application Platform SDK features additional runtimes such as Open ESB, Portlet Container, and Sun Java System Access Manager.

4.1 Highlights

(1) First robust, commercial, compatible Java EE 5 implementation. It is free for development, deployment, and redistribution.

(2) Ease of development with major revamp of programming model.

(3) EJB 3.0 support for POJOs means less to learn, less to code, and less to maintain.

(4) New Java Persistence API makes object-relational mapping cleaner and easier.

(5) New and updated web services (JAX-WS 2.0 and JAXB 2.0) simplifies SOA implementation.

(6) JavaServer Faces 1.2 facilitates building Web 2.0 applications with Ajax.

(7) Higher throughput, faster response time, and improved management features to streamline deployment.

(8) 30 percent faster startup time with 30 percent less memory

(9)Web services performance increased by up to 5 times.

Improved web services management.

(10) Integrated with Sun's NetBeans open source IDE and supports an Eclipse plug-in, offering developers the choice of complete end-to-end development and runtime environment.

(11) Provides a visual SOA development and deployment environment with integrated NetBeans environment.

(12)Integrated composite application support through JBI and BPEL

The code for Sun Java System Application Server 9.1 is 100 percent derived from open-source Project GlassFish.

As the industry's first compatible implementation of the Java Platform, Enterprise Edition (Java EE) 5 specification, the Java EE 5 SDK provides the foundation for delivering enterprise-class application services and web services. It offers a unique modular architecture based on some of the industry's most proven, high-performance, and standards-compliant components. Sun Java System Application Server 9.1 is designed for developer productivity with tools to help deploy applications quickly. With a small-footprint adoption of Java EE platform that is suitable for broad adoption and embedding in third-party systems and applications, Sun Java System Application Server 9.1 brings Java EE 5 technology into volume markets. It is completely free of license fees for development, deployment, and redistribution, with support available for an additional charge. Sun Java System Application Server 9.1 is the default container for NetBeans.

4.2 Java EE 5 Platform Support

With a primary focus on ease of development, the Java EE 5 platform offers developers ready access to a secure, portable, and scalable platform for their enterprise applications. Java EE 5 technology makes coding simpler and more straightforward. Sun Java System Application Server 9.1 supports all the technologies required by the Java EE 5 specification. These include but are not limited to:

Java annotations can now be used to specify the mapping of Java business objects to a relational database. The persistence implementation in the application server supports both the generation of database schemas from Java objects and the mapping of existing database schemas to Java objects. In addition to using persistence inside the EJB container, one can use the Persistence API directly with Java SE. This allows maximum flexibility in the application Enterprise JavaBeans (EJB) 3.0 that includes two new sets of APIs: a) A simplified EJB API which promotes ease of development, b) A new API for management of persistence and object-relational mapping.

Java annotations can now be used to specify the mapping of Java business objects to a relational database. The persistence implementation in the application server supports both the generation of database schemas from Java objects and the mapping of existing database schemas to Java objects.

In addition to using persistence inside the EJB container, one can use the Persistence API directly with Java SE. This allows maximum flexibility in the application persistence needs, with a common solution for both client and server environments.

JavaServer Faces 1.2 technology simplifies the building of user interfaces for web-based applications through its well-defined component, state, and event framework. JavaServer Faces technology works extremely well with Ajax technology, providing encapsulation to handle browser differences and to hide the complexities around JavaScript. Java BluePrints covers key issues and solutions for common problems encountered during the design and building of an Ajax application on the Java EE platform. Several JavaServer Faces and Ajax components that work with Sun Java System Application Server 9.1 are included as part of the catalog JavaServer Faces 1.2 technology simplifies the building of user interfaces for web-based applications through its well-defined component, state, and event framework. JavaServer Faces technology works extremely well with Ajax technology, providing encapsulation to handle browser differences and to hide the complexities around JavaScript. Java BluePrints covers key issues and solutions for common problems encountered during the design and building of an Ajax application on the Java EE platform. Several JavaServer Faces and Ajax components that work with Sun Java System Application Server 9.1 are included as part of the catalog.

4.3 Improving Developer Productivity

The Java EE 5 SDK increases developer productivity with the implementation of the new simplified Java EE 5 APIs and annotations that reduce significantly the amount of code a developer needs to write. Sun Java System Application Server 9.1 is included as part of the Java EE 5 SDK bundle with which developers can quickly learn, develop, and deploy new enterprise Java technologies. More than 6 million developers have downloaded earlier versions of the Java EE SDK. The Java EE 5 SDK also bundles:

BluePrints and Samples guidelines for developing Web 2.0-based applications with Ajax and other popular technologies; the Open ESB runtime (JBI, BPEL engine, and SOAP HTTP binding) that provides developers with a runtime environment for constructing complex composite applications. In addition, Sun Java System Application Server 9.1 is bundled with Java DB based on Apache Derby database, hence making it possible to develop and deploy out-of-the-box, end-to-end Java EE applications.

Sun Java System Application Server 9.1 also extends ease of deployment through an option for faster startup (on-demand initialization), and reduces the memory requirements.

NetBeans IDE 6.1 supports development of Java EE 5 applications, including web modules and Enterprise JavaBeans (EJB) 3.0 modules, and deploys to the Sun Java System Application Server 9.1. Sun Java System Application Server 9.1 also supports an Eclipse IDE plug-in that gives developers a choice in development environments, though this means that many advanced features of

NetBeans (notably Java EE 5 application development) will not be supported by Eclipse.

4.4 Foundation for SOA

Java EE 5 SDK is an ideal platform to develop SOA applications. Apart from Sun Java System Application Server 9.1 and NetBeans, Java EE 5 SDK bundles the Open ESB runtime, hence complementing cutting edge Java EE 5 technologies for building SOA components, managing a web services stack and providing a framework for a federated identity management system. Sun Java System Application Server 9.1 is integrated with the Metro web services stack that supports the latest web services standards that make implementing SOA easier, including JAX-WS 2.0 that specifies a web services API for the Java platform, and JAX-B 2.0 that specifies Java and XML binding. With JAX-WS 2.0 the code developers need to write can be significantly reduced. The Open ESB runtime contains JBI, a BPEL engine, and SOAP HTTP binding, and provides developers with a runtime environment for constructing complex composite applications. In other words, it provides an open and extensible architecture for collaboration between integration technology and web services in a SOA. The Open ESB runtime will help streamline development work for platform providers, tools vendors and systems integrators while future proofing customers' investments and ending vendor lock-in to proprietary integration architecturesSun Java System Application Server 9.1 works with Access Manager and Liberty SSO , hence providing a framework for a federated identity Management system. This allows individuals to use the same user name, password, or other personal identification to sign on to the networks of more than one enterprise in order to conduct transactions. In this manner, business partners can share applications without needing to adopt the same technologies for directory services, security, and authentication. Sun Java System Application Server 9.1 can connect to and work with a number of web services registries. A web services registry allows an enterprise to track and manage large and increasing numbers of web services that they are offering. By providing the ability to connect to the registry, Sun Java System Application Server 9.1 addresses both issues of web services access and SOA governance.

NetBeans IDE 6.1 provides SOA visual design tools for architects and programmers. XML Schema tools are used to visualize and edit XML Schema, and visualize the relationships between Schema elements. The tool for web services orchestration is used to author, build, deploy, and test BPEL processes.

4.5 Open Source and Project GlassFish Community

In June 2005, Sun launched the Project GlassFish community with the goal of developing a free, open-source, commercial-grade application server that implements the newest features of the Java EE 5 platform and related enterprise technologies. Open source accelerates the development and distribution of new features and ensures continued quality of the product by allowing many eyes to look at the code. Opening the development process enables more developers to work with the platform and makes developers more productive.

The code for Sun Java System Application Server 9.1 is derived from Project GlassFish. GlassFish is released under an OSI approved license (CDDL). Members participate in this robust community by exchanging information through a discussion forum and mailing list, filing defects and request for enhancements in the public issue tracker, and proposing changes to the source code.

The Aquarium, a group blog, collects news from and about the GlassFish community and offers tech tips from a variety of sources. Application Server-related add-ons, plug-ins, and technology projects can also be found on the GlassFish project web site.

4.6 Web Services Management

Web services are first-class manageable objects in Sun Java System Application Server 9.1. Web services deployed to the Application Server are thus automatically discovered and can then be managed and monitored. If monitoring is enabled for a web-service endpoint, information such as response time, throughput, and number of requests and faults is collected and can be viewed through the admin GUI. SOAP message content can also be examined. Meanwhile, a web-service testing page can automatically be generated—eliminating the need for explicit web-service client development. Web service can be virtualized by applying XSLT transformation rules to request/response. Web services audit, message-level security, governance (publish/unpublish to registry) and self management is supported. (Also see the article, "Managing and Monitoring Web Services in Project GlassFish.")

4.7 Call Flow Monitoring

Sun Java System Application Server 9.1 can be configured to monitor an incoming request as it flows through various containers in the application server and through the user application code. For example, the Application Server can break down the time spent in the web container, web application code, EJB container, and EJB application code. The collected information is stored in a database and is then available for query and analysis. When filtering is enabled in call flow, Application Server generates call flow data for only those requests that match the criteria—host or user id.

Afterwards, you can inject requests into a running system—specifically to generate trace information. You can then watch your requests flow through the system and evaluate the performance of specific types of requests in the context of normal load, bypassing requests from other sources.

4.8 Self-Management Rules

With the powerful and flexible infrastructure of Sun Java System Application Server 9.1, it is possible to automate application server management tasks by setting up a self-management rule. A self-management rule consists of an event and an action. Examples of events include message logging, monitoring threshold, timer, and JMX notifications. Actions are pieces of logic defined by the user, which are then encapsulated in JMX managed beans (MBean). When an event that matches one of the events defined in a self-management rule is triggered , the associated action is executed. For example, an administrator might define a management rule such that he or she receives email when a SEVERE message is logged in the application server.

4.9 Java Web Start Software

In addition to defining server-side components like EJB components and servlets, the Java EE platform defines application clients. Typically, these are rich UI applications that run on client machines and connect to the Application Server for retrieval and processing of backend data. With Sun Java System Application Server 9.1, users can deploy a Java EE application client to the Application Server and then take advantage of Java Web Start software for distribution of the application to client machines. By visiting a single URL in a user's browser, Java Web Start software transparently retrieves and installs the bits necessary to run the Java EE application client. The downloaded bits are cached and can be reused in subsequent sessions. That means no more manually copying and installing the application client bits on individual client machines.

4.10 Performance

In addition to startup, memory footprint, and deployment performance improvements, Sun Java System Application Server 9.1 features increased runtime performance. The server includes a highly scalable HTTP connection handler that is implemented with lower-level Java NIO primitives and that can handle thousands of connections with a small number of threads. Sun Java System Application Server 9.1 supports the use of the Fast Infoset standard to reduce the size and processing time of XML and SOAP messages. Depending on the size of the XML messages, processing time is 3 to 5 times faster, and message size is 1.3 to 5 times smaller. See Sun's Fast Infoset article for more details. Sun Java System Application Server 9.1 also provides 64-bit support in the Solaris operating system, so the server benefits from more than 4 gigabytes of virtual address space.

4.11 Changing the Economics of Application Servers

Sun Java System Application Server 9.1 enables enterprises to standardize on the new Java EE 5 compatible application server for developing and deploying enterprise-class applications and services—without incurring product license fees. The small footprint and free development and deployment licenses also make it extremely suitable for bundling and distribution with Java EE applications, subject to Sun’s current terms and conditions. Unlike other commercially available application servers, Sun Java System Application Server is built on the very platform that defines the Java EE standard—Java EE 5 Reference Implementation source code. The same development team that delivers the Reference Implementation developed Sun Java System Application Server 9.1. This assures the most rigorous Java EE standard compliance and web services interoperability through support of the WS-I Basic Profile. Enterprises and application vendors benefit from this Java EE standard-focused approach because it reduces the risks of proprietary lock-in, and enables compliant applications that are portable across compliant application servers—without costly modifications. Sun Java System Application Server is designed to help enterprises and service providers maximize their freedom of choice through Java technology’s Write Once, Run Anywhere™ simplicity and is optimized for web services. It helps enterprises and service providers lower their total cost of ownership, accelerate time to market, and increase productivity.

4.12 Sun Java System Application Server 9.1 FC

The Application Server builds on the quality and feature-rich Beta 2 to provide greater value-add features for the enterprise. The Application Server 9.1 features include clustering, in-memory replication, more improvements to the Grizzly-based HTTP engine, enhanced administrative functionality, improved self-management capabilities, update center functionality that allows installation and/or updates of additional components, and much more Application Server Core Services.

Sun Java System Application Server is the first free Java EE 5 compatible application server for development, deployment, and redistribution. Supports the Enterprise JavaBeans (EJB) 3.0 that includes two new sets of APIs: a) A simplified EJB API which promotes ease of development, b) A new API for management of persistence and object-relational mapping. Supports Java annotations that can be used to significantly improve the developer productivity.

Includes JavaServer Faces 1.2 APIs for building powerful GUIs for Java EE applications. Offers new capabilities to develop Web 2.0 and Ajax applications. It now supports the blueprints for Ajax that allows development of impressive dynamic client side web applications. Is included as part of the Java EE 5 SDK bundle. Other components of the Java EE 5 SDK are the BluePrints and Samples guidelines; the Open ESB runtime that includes JBI, BPEL engine, and SOAP HTT, and Java DB based on Apache Derby database.

Extends the ease of deployment through an option for faster startup (on-demand initialization), and also reduction in the memory requirements.

Offers support for NetBeans, and Eclipse (through a plug-in).

Automatically discovers, and then manages and monitors web services that are deployed to the Application Server.

Offers 3 to 5 times improved web services performance. This is achieved through implementation of Fast Infoset technology.

Can monitor an incoming request as it flows through various containers in the application server and through the user application code.

Can automate application server management tasks by setting up a self-management rules that monitor events as they occur, and trigger an appropriate action.

Includes a Generic Resource Adapter. This will ensures support for IBM MQ Series.

Offers comprehensive support for web services, and support for WS-I Basic Profile 1.0 ensures interoperability.

Lowers TCO through rigorous compliance with the latest Java EE 5 standard and is free for development and deployment.

Allows existing applications to become new web services through integrated support of SOAP and WSDL.

Supports database connectivity to Oracle, Sybase, IBM, Microsoft SQL Server, MySQL, and Derby.

Supports various security standards: Single Socket Layer (SSL)v2, SSLv3, Transport Layer Security (TLS) 1.0, X.509 certificates, PKCS #11, FIPS-140, and 168-bit step-up certificates.

Offers enhanced and easy web-based administration, as well as full administration from a command-line interface.

Provides 64-bit support in the Solaris operating system, so the server benefits from more than 4 gigabytes of virtual address space.

4.13 Java EE 5 Development Environment

Support for new standardized Java EE 5 deployment APIs.

Deploys Java EE 5 Connector Architecture-compliant connectors.

Supports NetBeans IDE 6.1 software integrated development environments.

Web Services Metadata for the Java Platform 1.0.

Java API for XML-Based Web Services (JAX-WS) 2.0.

Java Architecture for XML Binding (JAXB) 2.0.

Streaming API for XML (StAX) 1.0.

Supports third-party Java EE software integrated development environments, such as Eclipse and Borland Jbuilder.

4.14 Serious Software Made Simple

Sun provides a complete portfolio of affordable, interoperable, and open software systems designed to help maximize the utilization and efficiency of the enterprise IT infrastructure. Built from the secure, highly available foundations of UNIX and Java, these systems deliver implementations that are preintegrated and backward compatible. Sun’s portfolio consists of Solaris and Linux software for SPARC® and x86 platforms, the N1 platform for dynamic and utility computing, and the Sun Java System—five integrated software systems for the data center, the desktop, the developer, mobile devices, and smart card identity implementations.

The Java System is a radical new approach that changes forever the way businesses acquire, develop, and manage software. Only Sun has the experience and the end-to-end portfolio to deliver such a unique and industry-revolutionizing strategy. With the Java System, network services and critical business applications are up and running faster, easier, and at a lower cost than ever before, so the enterprise can focus on innovation, competition, and bottom-line results.

5. The system realization of stock supervision based on J2EE

Enterprise applications are large-scale business applications that are complex, data-centric, and mission critical. Their complexity is often based on their encapsulation of the intricate business processes, rules, and standards of the application domain for which they are created.

Stated simply, these are applications that companies rely on for the smooth functioning of their business.

5.1 Main technologies of the supervision system

The system of stock supervision includes Page of client layer, Data model layer and Transaction controlling layer. The page of client adopts JSP2.0, the data model layer adopts the new EJB3.0 of Hibernate framework, EJB3.0 obviously reduces the complexity of the bottom development, Transaction controlling layer use the Struts1.29（MVC）. The editor of Java development kits is JDK5.0, Web application server adopts Tomcat 5.5,the database server uses MySql5.0.

The main structrue of Hibernate is shown in Figure 3.

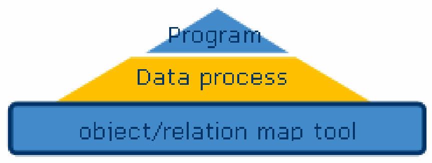


Fig. 3 The structrue of Hibernate

The characteristics of Hibernate is automation, code economization, data types-independent, what we nee is that creates a copy of the documents of XML, the document show the class of the database and the relation of the database table and this class, then we obtain data as form of object or we save object as data.

5.2 The realization of Struts

Struts is a implement of the Model-View-Controller Struts by servlet and JavaServer Pages technologies. As in the following Fig.4.

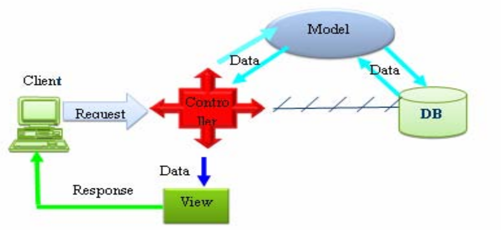


Fig. 4 The framework of Struts

5.3 The design and test of the stocks invest supervision

5.3.1 Data Model

Java EE program need a set of data model and persistence of entity bean because Entity Bean is connected with EJB, Entity is included inside container, so this result in repeat ,and repeat is nightmare of the program maintenance. Java Persistence API solves this problem. In JPA, Entity in JPA may is away from Persistence Context as Java class. Data model is show in Fig.5.

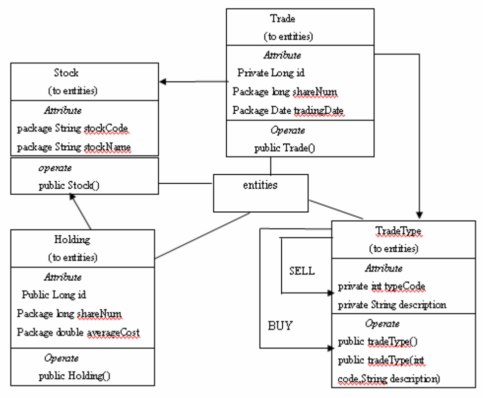
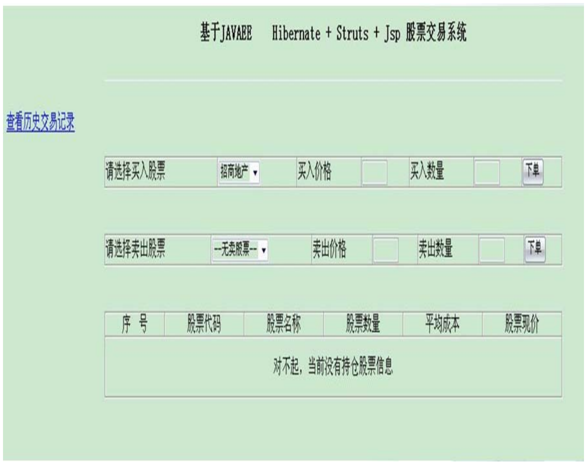


Fig. 5 Data model

5.3.2 Test of the System

The system primitive page of stocks invest supervision is shown as Fig.6.



基于JAVAEE Hibernate + Struts + Jsp 股票交易系统

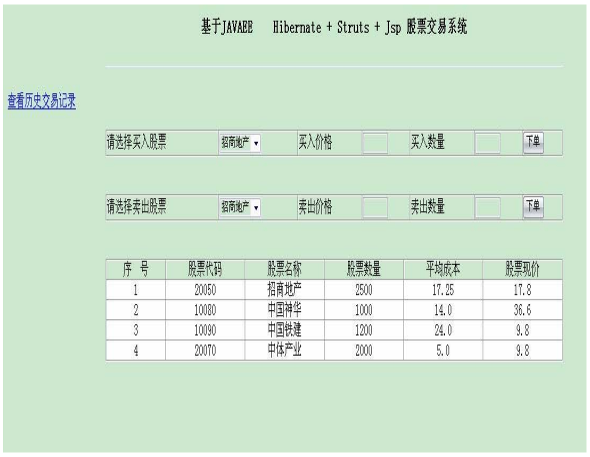
查看历史交易记录

请选择买入股票	招商地产	买入价格		买入数量		下单
请选择卖出股票	--无卖股票--	卖出价格		卖出数量		下单

序　号	股票代码	股票名称	股票数量	平均成本	股票现价
对不起，当前没有持仓股票信息					

Fig.6 Primitive page

The system of bargaining process is shown in Fig. 7.



基于JAVAEE Hibernate + Struts + Jsp 股票交易系统

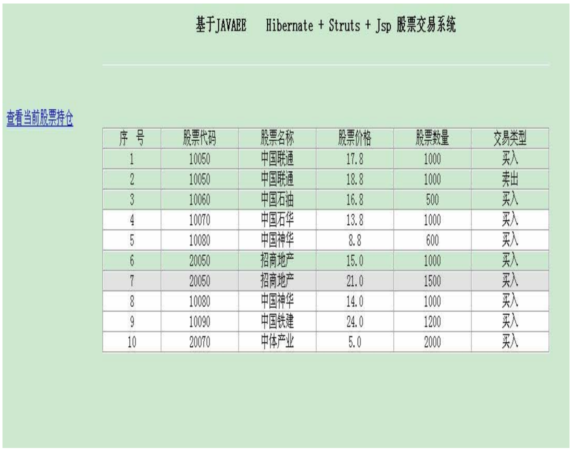
查看历史交易记录

请选择买入股票	招商地产	买入价格		买入数量		下单
请选择卖出股票	招商地产	卖出价格		卖出数量		下单

序　号	股票代码	股票名称	股票数量	平均成本	股票现价
1	20050	招商地产	2500	17.25	17.8
2	10080	中国神华	1000	14.0	36.6
3	10090	中国铁建	1200	24.0	9.8
4	20070	中体产业	2000	5.0	9.8

Fig.7 The bargaining process

The system of transaction record query is shown in Fig.8.



基于JAVAEE Hibernate + Struts + Jsp 股票交易系统

查看当前股票持仓

序　号	股票代码	股票名称	股票价格	股票数量	交易类型
1	10050	中国联通	17.8	1000	买入
2	10050	中国联通	18.8	1000	卖出
3	10060	中国石油	16.8	500	买入
4	10070	中国石华	13.8	1000	买入
5	10080	中国神华	8.8	600	买入
6	20050	招商地产	15.0	1000	买入
7	20050	招商地产	21.0	1500	买入
8	10080	中国神华	14.0	1000	买入
9	10090	中国铁建	24.0	1200	买入
10	20070	中体产业	5.0	2000	买入

Fig.8 The transaction record query

6. Java Security

The Java platform was designed with a strong emphasis on security. At its core, the Java language itself is type-safe and provides automatic garbage collection, enhancing the robustness of application code. A secure class loading and verification mechanism ensures that only legitimate Java code is executed.

The initial version of the Java platform created a safe environment for running potentially untrusted code, such as Java applets downloaded from a public network. As the platform has grown and widened its range of deployment, the Java security architecture has correspondingly evolved to support an expanding set of services. Today the architecture includes a large set of application programming interfaces (APIs), tools, and implementations of commonly-used security algorithms, mechanisms, and protocols. This provides the developer a comprehensive security framework for writing applications, and also provides the user or administrator a set of tools to securely manage applications.

The Java security APIs span a wide range of areas. Cryptographic and public key infrastructure (PKI) interfaces provide the underlying basis for developing secure applications. Interfaces for performing authentication and access control enable applications to guard against unauthorized access to protected resources. The APIs allow for multiple interoperable implementations of algorithms and other security services. Services are implemented in providers, which are plugged into the Java platform via a standard interface that makes it easy for applications to obtain security services without having to know anything about their implementations. This allows developers to focus on how to integrate security into their applications, rather than on how to implement complex security mechanisms.

The Java platform includes a number of providers that implement a core set of security services. It also allows for additional custom providers to be installed. This enables developers to extend the platform with new security mechanisms

7. Conclusion

Java EE 5.0 has many convenience and of JDK 5.0 ,furthermore Java EE 5.0 has superiority in data persistence and EJB programming. The system model of stocks invest supervision has implemented basic functions, but the system will get along with transaction security and mobile payment.

References:

[1] DeChao LI, J2EE Application Framwork Realizaton Based on Open Project[D],Southeast University, 2006
[2] JingZhou FU, Master Hibernate 3.0------Java Database Persistence Layer Development Practice[M], Posts & Telecom Press, 2007
[3] http://www.Hibernate.org/
[4] Ted Husted,Cedric Dumoulin,George Franciscus,David Winterfeldt. Struts in action .Greenwich: MANNING
[5]http://tech.ccidnet.com/art/1060/20040615/121091_1.html
[6] Java 2 Platform Enterprise Edition Specification,v1
[7] R. Anderson, “Trusted Computing Frequently Asked Questions,” August 2003, http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html.
[8] Shamir, “How to share a secret,” *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.
[9] Y. Lindell and B. Pinkas, “Privacy preserving data mining,” *Journal of Cryptology*, vol. 15, no. 3, pp. 177–206, 2003.
[10] N. Hu and S.-C. Cheung, “A new security model for secure thresholding,” in *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP '07)*, Honolulu, Hawaii, USA, April 2007.
[11] H. Lipmaa, “Oblivious Transfer or Private Information Retrieval,” University College London, http://www.adastral.ucl.ac.uk/~helger/crypto/link/protocols/oblivious.php.