Using Java Technologies in Developing Enterprise Systems

Safwan Al Salaimeh and Amer Abu Zaher
1,2 Computer Science Department Irbid National University Irbid, Jordan.

Abstract: Many computer languages, environments, and tools are exist and available for developing corporate systems. Each has its own strengths, weaknesses, and specific focus in terms of the types of solutions it is best suited for. An corporate systems must have a technology environment diverse enough to provide the variety of solutions required without creating an excessive burden. Java technologies and tools can be rated to assist corporate system development.

Key words: Data processing, corporate systems, distributed process data, distributed database systems, client application, information environment.

INTRODUCTION

The increasingly growth of the web sites (corporate system) as a major source of resource storage and retrieval has led to storage of information in a reliable database. The main problem is the number of transactions that can be supported by a database system i.e., the number of hits it can handle. This has led to the use of many sources of data that can be accessed. In a Distributed Database System (DDS), the data is stored in regions that are distributed geographically and can be accessed either locally or remotely.

Many applications have been employed to develop corporate systems. One of these is Java technology. Two components Exist: Java and the universal client, known as the browser. Java environment makes it almost impossible to introduce programming errors of the kind which plagued other languages. There is no explicit requirement to free allocated memory as memory is automatically reclaimed when no longer referenced. Similarly it is impossible to reference a variable which has not been initialized.

Java Environment:

The object orientation and strong typing features it has, contribute to develop a "clean" code, no typical bugs. Programs will not require extensive debugging to uncover errant pointers or uninitialized variables. The Java compiler will fail to compile code which contains these errors.

Java provides robust exception handling, insisting that code either handle or pass along exceptions which could be thrown. Run-time exceptions are rare, but cause a detailed stack-trace to be generated which will assist in problem determination efforts.

Three different environments available for Java: applets, servlets and full applications. The applet class is familiar of using a web browser recently. Full applications are still fairly rare but increasingly become common for small, special-purpose enterprise applications. The newest Java environment is the servlet, an application which runs on a web server. Supported on all the major server products, servlets can replace CGI scripts or serve as elements of an n-tier architecture.

While powerful, not necessarily that Java applets are the best way to interact with end users. In order to load, a large applet can require considerable network bandwidth, and time. sophisticated users who ask for fast response to requests will not accept that.

A more compact interface model is provided by HTML forms. Used in integration with SSL, forms can both collect and present information from/to user. While forms processing is currently supported by CGI scripts, a more strong and better choice alternative is the Java servlet.

The environment of Java has become the basic choice for powerful, large-scale enterprise systems and is becoming popular in embedded systems and portable computing devices, as well. Faster processors and less expensive mass storage have allowed Java to transcend the language's original limitations, and Java's platform independence makes it suitable for developing web services and global enterprise applications. Additionally, many smaller systems and even hand-held devices can benefit from applications written in Java.
Distributed Corporate Systems:

The corporate systems integrate internal applications, such as database management, document management, and e-mail, with external applications, such as news services and customer Web sites. It is a Web-based interface that gives users access to all these applications through their PCs. Enterprise portals bring both external and internal information to all employees’ desktops. They also enable collaboration (Deborah Nightingale, 2009). The corporate system is said to be distributed when a number of geographically-distributed computer powers used in processing data. The distributed data processing, as applied to computer network, means technology when the application programs running on one network node provide substantial process and representation of data, and another one network node with database server on it realizes data access and select operations. Besides the distributed process data, a geographically-distributed data warehouses exist, for example when a different databases, tables or their different parts are stored on a different network nodes. At the same time it is necessary during any method of data fragmentation to make the fragmentation clear for data processing. In other words user does not need to know the exact data allocation (Robert Orfali, 1998).

The state of design tools distribution determines so that the workstation usually connected to a network and a modern application program represents then control script but not the monolithic piece of code. And this script puts in action a number of geographically-distributed objects, which exist like applets, servlets, scriptlets and other design structures. Using such kind of design tools in banking information systems and e-commerce systems will greatly grow up in the near future because their divisions may function on different hardware platforms, heterogeneous computer networks, under various operation systems, using various programming languages and remote access technologies (Robert Orfali, 1998; Maghrai Thakkar., 2000; White, 2002; Eric Armstrong, 2005).

The Client-server Technology:

The “client-server” technology is the most of working corporate systems at present time use. There are three basic parts in client-server models: user interface, which displays the information, realizes graphical user interface functions and forms queries to server in compatible with it format; functional logic, which realizes necessary computing, logical comparisons, additional data retrievals and business rules, typical for concrete application program; database, which executes data retrievals, modifies data and process them in compliance with received commands. Depending on these component location methods on client workstation and server are possible 2-tier, 3-tier and n-tier client-server corporate systems architecture models.

2-tier models can be: with intellectual client, when the client workstation contains user interface and functional logic, and the server contains – database; with intellectual server, when the client workstation contains user interface, but the server contains functional logic and database. The last model is better protected, has lower communication channel traffic, the transaction realization is good enough and provides user rights restriction properly.

3-tier models with distributed services provide independent functional relationship between user interface, functional logic and database. User interfaces and smaller part of functional logic are situated on the client workstations. The greater part of functional logic is situated on the application (middleware) server. The database is situated on the database server. The applications in the model with distributed services are independent; they interact through the computer network with the application server, which interacts with the database server as needed.

Object-oriented network models of distributed services join the distributed databases models and the distributed process data. The software of such kind models consist the aggregate of object units, interactive between each other through the computer network using the standard interfaces. This approach lets use of units over and over again and more economically use computer resources. In this model each object depending to situation can be server or client. The object computing architecture based on the distributed network services represents a new highly developing division of computer technologies which is widely uses for distributed corporate systems development (Mike Morgan., 1999). The distributed process data is extremely perspective for dynamic Web-oriented applications development. The theme of “super thin” clients realization is closely concerned with them, when is using only the browser medium for execution of client application programs. The effectiveness of such kind of a technology for corporate system development is explained by easiness of HTML-browser realization for any operating system. If we take into account that the cost of servicing for one server and cost of servicing for a thousands connected “thick” clients to it are not comparable then we can make a conclusion – successfully realized corporate system is sharply bringing down the overhead charges to service and maintenance.
**The Technology of Java2:**

According to the requirements of developing corporate systems, which spans devices ranging from large servers to small mobile devices, a Java implementation will need to leverage many APIs from across the Java platform. Three editions of Java2 exist; Java 2 Standard edition (J2SE), Java 2 Enterprise Edition (J2EE) and Java 2 Micro Edition (J2ME). The reason for this separation of editions is that the range of possible Java operating contexts, and the APIs to support them, is so wide that a single set of standard APIs for all types of application and architecture would be excessively large and would include much redundancy. The number of optional packages that spanned all types of system would also be very large. The standard edition is itself divided into core Java (all the fundamental APIs) and desktop Java (the rich client APIs), while the enterprise edition is essentially a superset of the standard edition core. All of the APIs of the standard edition could conceivably be used in enterprise development, and both can use a standard Java Virtual Machine (JVM). The Micro edition, however, comprises various subsets of the core with many additional specialized APIs, and requires customized JVMs.

The Java2 specification provided good choices. Enterprise Java Beans EJBs had remote interfaces and could be used only in distributed applications. Remote Method Invocation (RMI) was the only choice for supporting remote clients.

**Two Developments Exist-by Implications for Java2 Design:**

- The EJB 2.0 specification allows EJBs to have local interfaces with/without remote interfaces. Through local interfaces, EJBs can be invoked by components in an integrated J2EE application running in same JVM: for example, components of a web application.
- The existence of the XML-based Simple Object Access Protocol (SOAP) as a widely accepted, platform-agnostic standard for RMI, and widespread support for web services.
- The integration of EJB local interfaces and web services, EJB can be used without RMI, and support remote clients without EJB. This gives us much greater freedom in designing Java to develop enterprise systems. The key goal of the EJB specification is to simplify application code "The architecture of EJB will easily write applications; developers' Application will not need to understand low-level transaction and state management details, multi-threading, connection pooling, and other complex low-level APIs.". The following consequences of Using EJB technology must be undertaken: harder to test and deploy applications and Using EJB with remote interfaces may hamper practicing OO design.

There are many reasons which strongly consider the use of EJB:

- Remote access permission to application components.
- Distribute application components across multiple servers.
- Integration of Java and/or CORBA.
- Implementing message consumers when an asynchronous model is appropriate.

EJB's simplification of multi-threaded code is a strong, but not decisive, argument for using EJB. The availability of declarative transaction management via container-Managed Transactions CMT is the most compelling reason for using EJB. EJBs are a good solution to problems of distributed applications and complex transaction management. However, many applications don't encounter these problems. EJBs add unnecessary complexity in such applications. An EJB solution can be likened to a truck and a web application to a car. When we need to perform certain tasks, such as moving large objects, a truck will be far more effective than a car, but when a truck and a car can do the same job, the car will be faster, cheaper to run, more maneuverable and more fun to drive (Rod Johnson, 2002).

Data access technology must be considered to decide whether to use EJB or not. Data access strategy often determines the performance of enterprise systems, making it a crucial design issue. Regarding data access, developers of Java2 are inflexible. Some assumptions are rarely challenged: Portability between databases is always essential and the best choice when working with relational databases is always Object/Relational (O/R) mapping. Whatever the data access strategy used, it is preferable to decompose business logic from the details of data access, through an abstraction layer. An entity bean represents a business object in a continuous storage mechanism (Rod Johnson, 2002). An examples of business objects are customers, orders, and products. Entity beans are inadequate to implement this design principle. It can be used just to isolate data access code. Unfortunately, entity beans will do this, with a high runtime overhead. Entity beans can be widely used in the EJB container and in a O/R mapping technology. Linking data access to the EJB reduces architectural flexibility and hard to test the applications. Entity beans are still under-specified, despite enhancements in EJB. This causes it difficult to use them for solving many common problems. They often lead to inefficient use of relational databases, resulting in poor performance.
Development of Enterprise Systems:

Dividing enterprise systems into multiple tiers by Experience has shown the value of cleanly. Enterprise systems are reflected by the three-tier architecture of Java2 in a wide range which have proven more scalable and flexible than client server systems, in which there is no middle tier.

Enterprise Information Edition EIS-tier consists of enterprise resources that must be accessed to accomplish Database Management Systems (DBMSs) and Main frame application. Although the server manages transactions and connection pooling, the EIS tier is outside the control of the Java2 server. the EIS tier resources may impact on the implementation of the middle tier. A Java2 server is responsible to contribute in the connections to EIS resources, transaction management across resources, and to ensure that the Java2 application will not affect on the security of the EIS system.

The middle tier consists of application's business objects, and mediates access to EIS tier resources. The component of this tier benefit most from Java2 container services such as transaction management and connection pooling. Middle-tier components are independent of the chosen user interface. We can split the middle tier into EJBs, and objects that use the EJBs, to support the interface. If we use EJB.

The User Interface tier UI exposes the middle-tier business objects to users. In web applications, the UI tier consists of servlets, helper classes used by servlets, and view components such as JSP pages. For clarity, we'll refer to the UI tier as the "web tier" when discussing web applications.

Conclusion:

The realized practical scientific analysis of possibilities and quality of modern technologies Java, lets make conclusion that these technologies bring together the most progressive methods of object-oriented programming, visual component design and network computing decision making.

REFERENCES

Deborah Nightingale, Massachusetts Institute of Technology 2009. "Principles of Enterprise Systems" Published and used by MIT ESD and CESUN with permission.


