# NAVAL POSTGRADUATE SCHOOL
## Monterey, California


# THESIS

**ANALYSIS, DESIGN AND IMPLEMENTATION OF A WEB DATABASE WITH ORACLE 8**

by

Ugur Demiryurek

March 2001

| | |
|---|---|
| Thesis Advisor: | Thomas Wu |
| Thesis Co-Advisor: | Chris Eagle |

**Approved for public release; distribution is unlimited**

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*



| 1. AGENCY USE ONLY *(Leave blank)* | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | March 2001 | Master's Thesis |

| 4. TITLE AND SUBTITLE | 5. FUNDING NUMBERS |
|---|---|
| Analysis, Design and Implementation of a Web Database with Oracle8i | |
| **6. AUTHOR(S)** | |
| Demiryurek, Ugur | |
| **7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)** | **8. PERFORMING ORGANIZATION REPORT NUMBER** |
| Naval Postgraduate School<br>Monterey, CA 93943-5000 | |
| **9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)** | **10. SPONSORING / MONITORING AGENCY REPORT NUMBER** |

**11. SUPPLEMENTARY NOTES**

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

| 12a. DISTRIBUTION / AVAILABILITY STATEMENT | 12b. DISTRIBUTION CODE |
|---|---|
| | |

**13. ABSTRACT** ***(maximum 200 words)***

This thesis represents a model of web-database analysis, design and implementation. An electronic bulletin board for the Naval Postgraduate School is implemented for demonstration. The model includes Oracle8i DBMS as the database, Java (Java Server Pages, Java Script, Enterprise Java Beans, Java Servlets) as the programming language. Apache HTTP Server v 1.3 Tomcat v.1.2 is used as the Web server and JSP engine. Windows NT4.0 served as the OS environment. From the technical aspect, Database Management Systems, Web-Database Architectures, Server Extension Programs, Oracle8i, as well as several other software and hardware components are reviewed, and some are recommended.

| 14. SUBJECT TERMS | 15. NUMBER OF PAGES |
|---|---|
| Oracle DBMS, Oracle8i, Java Server Pages, Enterprise Java Beans, Web-Database, Apache/Tomcat1.2, Two-tiered Architecture, Multi-tiered Architecture | 112 |
| | **16. PRICE CODE** |

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFI- CATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| Unclassified | Unclassified | Unclassified | UL |

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298

THIS PAGE INTENTIONALLY LEFT BLANK

**Approved for public release; distribution is unlimited**

**ANALYSIS, DESIGN AND IMPLEMENTATION OF A WEB DATABASE WITH ORACLE 8I**

Ugur Demiryurek
Lieutenant Junior Grade, Turkish Navy
B.S., Turkish Naval Academy, 1995

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN COMPUTER SCIENCE**

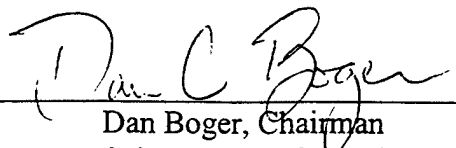
from the

**NAVAL POSTGRADUATE SCHOOL**
**March 2001**

Author: 
Ugur Demiryurek

Approved by: 
Thomas Wu, Thesis Advisor


Chris Eagle, Thesis Co-Advisor


Dan Boger, Chairman
Department of Computer Science and Information System & Technology

THIS PAGE INTENTIONALLY LEFT BLANK

## ABSTRACT

This thesis represents a model of web-database analysis, design and implementation. An electronic bulletin board for the Naval Postgraduate School is implemented for demonstration. The model includes Oracle8i DBMS as the database, Java (Java Server Pages, Java Script, Enterprise Java Beans, Java Servlets) as the programming language. Apache HTTP Server v.1.3 / Tomcat v.1.2 is used as the Web server and JSP engine. Windows NT4.0 served as the OS environment.

From the technical aspect, Database Management Systems, Web-Database Architectures, Server Extension Programs, Oracle8i, as well as several other software and hardware components are reviewed, and some are recommended.

THIS PAGE INTENTIONALLY LEFT BLANK

## ACKNOWLEDGEMENTS

I would like to thank my thesis advisor Prof. Thomas Wu, for introducing me to the basics of Database Management Systems, for his help in completing my thesis, and for his valuable instruction, which made this a worthwhile learning experience for me.

I also would like to thank Prof. Chris Eagle for his continuous support and encouragement throughout all phases of the thesis.

I would like to express my gratitude to Prof. Rex Buddenberg, who gave me an office and all the hardware that I needed to implement my thesis. I could not have set up the architecture without his help.

I would also like to thank Paul Dorsey (an author of several Oracle Press books), Todd Jonz (the founder of Infoseek), and Dr. Donald Bruce (CEO of Virtual Health Networks) for letting me join their development team and teaching me real world Enterprise Applications.

THIS PAGE INTENTIONALLY LEFT BLANK

## TABLE OF CONTENTS

# INTRODUCTION

I have always been curious to know how large Web sites are designed and constructed. Clearly, there must be some sort of underlying database system for these sites. But when I select "view source" in my Web browser, I see nothing but the HTML. How do these underlying databases work? What are the design issues? These are the questions that motivate this thesis.

I started the thesis with a lot of enthusiasm, as well as some uncertainty about how I would go about answering these questions. I knew basic things about static and dynamic HTML. I knew a little SQL code and ODBC/JDBC connection methods from my Introduction to Database class. I had already spent two quarters learning Java. I was eager to use what I had learned from my classes to implement an Enterprise Application, and I was ready for a hands-on experience. The final product is the "NPS Bulletin Board", which is intended to provide the Naval Postgraduate community with a convenient way to place advertisements for personal items on sale.

In my implementation I decided to use the following tools: Oracle8i DBMS as a database, Java (Jsp, Java Servlets, JavaScript, Enterprise Java Beans) as a programming language, Apache/Tomcat v.1.2 as a Web server and Jsp engine, and Windows NT 4.0 platform as an OS. The road was bumpy and uphill because I first had to learn how to install, configure and use Oracle and Tomcat. Second, putting all of these elements together and making an Enterprise Application by myself was not as easy as I had thought it would be.

In my thesis, I also present currently available technologies in web-database development. The outline of the thesis is as follows:

In Chapter II, I present an overview of database system concepts, as well as past and current Database Management Systems. Then, I discuss Relational Database Management Systems (RDMS), Object Oriented Database Management Systems (OODBMS) and Object Relational Database Management Systems (ORDBMS). I also concentrate on the modeling concepts of the Entity Relationship (ER) model, which is a popular high-level conceptual data model for database applications.

In Chapter III, I introduce and discuss the componenents and architecture of web databases in detail. First, I present two current architectures: the two-tiered client/server and the three-tiered web database. Second, I present Server extension programs such as cgi, asp, and jsp, among others. Third, I explain the database connection programs; ODBC and JDBC. Finally, I present the Client Side extension program.

In Chapter IV, I describe the Oracle8i architecture for web information system design. I mainly explore the two main approaches offered by Oracle: a two-tier architecture with Oracle WebDB, and a three-tier architecture using the Oracle Application Server (OAS).

Chapter V, my concluding chapter, presents my reasons for choosing Oracle and Java to implement the NPS Bulletin Board.

# II. DATABASE MANAGEMENT SYSTEMS

In this chapter, I will present a brief review of Database Management Systems. I will discuss early database systems and Relational Database Management Systems (RDBMS). I will introduce the Object Oriented Database Management System (OODBMS) and the Object Relational Database Management Systems (ORDBMS). I will concentrate on the modeling concepts of the Entity Relationship (ER) model, which is a popular high-level conceptual data model for database applications.

## A. EARLY DATABASE MODELS

In the days before the relational database model, two data models were commonly used to maintain and manipulate data: the Hierarchical Database Model (HDM) and the Network Database Model (NDM).

### 1. Hierarchical Database Model

In a hierarchical database model, data is structured hierarchically. This model can be easily visualized as an inverted tree. Relationships in a HDM are represented in terms of a parent and child. This means that in a HDM a single table will act as the root and other tables will act as the branches extending from the root. Therefore, a parent table can be associated with many child tables, but a child table can have only one parent table. These tables are explicitly linked via a pointer. To access any record in this model, the user always needs to begin from the root table and travel through the tree to the target.

One advantage of this type of database is that referential integrity is built in and automatically enforced. More precisely, a record in a child table must be linked to an existing record in a parent table; if a record is deleted in a parent table, all associated records are deleted in any child tables.

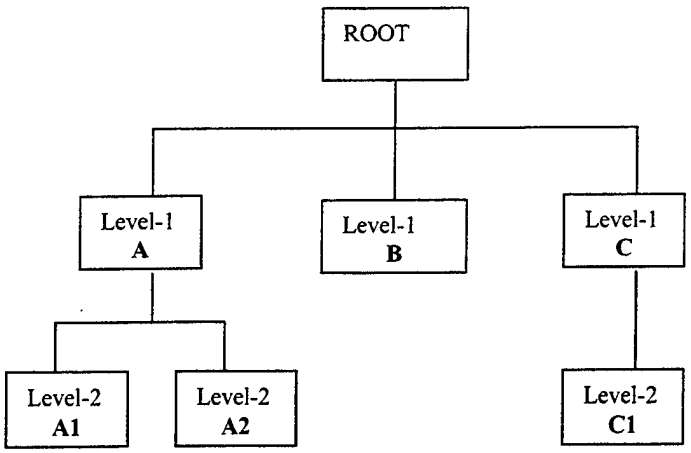



Figure 2.1 A Hierarchical Database Model Diagram

A problem occurs in a HDM when there is a need to store a record in a child table that is currently unrelated to any record in a parent table. However, if a dummy record is inserted in the parent table, the rules can be bent without breaking them. But this option is not optimal.

Another problem in a HDM is redundant data. Redundant data occurs because HDM is not flexible enough to allow many-to-many relationships. Therefore, the data should be repeated in both tables, participating in a many-to-many relationship.

The hierarchal model lent itself to the tape storage systems used by mainframes in the 1970's, and was very popular among companies that used those systems. But despite the fact that the HDM provided fast and direct access to data and was a useful model in a number of circumstances, it was clear that a new database model was needed to address the problems of data redundancy and complex relationships (many-to-many) among data. (Hernandez, M., 1998, pp. 11-12)

## 2. The Network Database Model

The Network Database Model (NDM) was developed to address some of the problems of the Hierarchical Model. As with the hierarchical model, the structure of the NDM can be visualized as an inverted tree. However, in this model there can be several inverted trees that share branches. Sharing branches solves the many-to-many relationship problem, which cannot be implemented in a HDM. Figure 2.2 shows a diagram of a NDM structure:

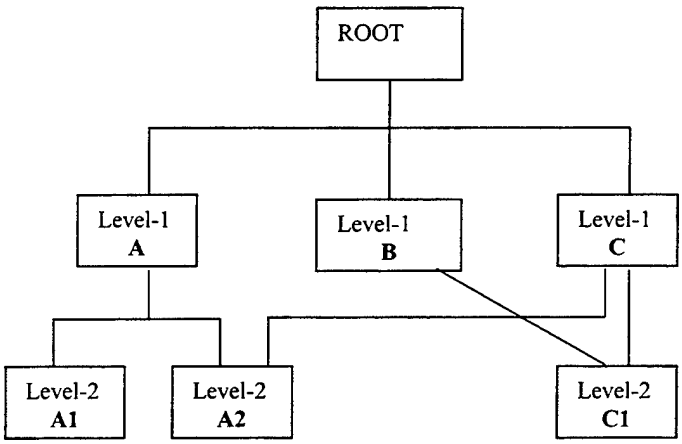



Figure 2.2 A Network Database Model Diagram

In NDM, all access to the data is defined by the pathways that link the schemas, which are the views of the database as they appear to an application program. The NDM also introduced the concept of a data definition as well as a data definition language(DDL). (Kroenke David M., 1998, p. 473)

The difficulty for the network model was that only defined relationships could be queried using defined paths in the schema. Any new relationship meant changing the schema of the database.

## B. RELATIONAL DATABASE MODEL (RDM)

During the late 1960's Dr. E. F. Codd, a mathematician, was looking in to new ways to handle large amounts of data. He had the idea that applying disciplines and structures of mathematics to data management would help to solve many of the problems encountered when applying other database models, such as data redundancy and weak data integrity.

Dr. Codd introduced the relational model of data in June 1970. He based his model on two branches of mathematic, set theory and first order predicate logic. The original papers by Codd did not receive much attention until IBM became interested in the relational database concept. IBM used his ideas to create a series of design documents for a relational database called System/R and a relational language, which they called Structured Query Language (SQL). (Hernandez, M., 1998, p. 21)

In RDM, the physical order of the records or the fields in a table is completely irrelevant. Each record in the table is identified by a field (called primary key) that contains a unique value. Therefore, a user is not required to know the physical location in order to retrieve its data. The relationship between the tables avoids data redundancy and duplication.

In RDM data is retrieved by specifying the appropriate fields and the table(s) to which they belong. One way of retrieving data is to use SQL. SQL is the standard language used to create, modify, and query relational databases.

Essentially, there are three basic components of the relational database model: relational data structure (tables), the rules that govern the organization of the data structures (constraints), and the operations that are performed in the data structures (inserts, deletes, and updates).

In summary, RDM has a number of advantages over the early models, such as

- **Built-in multilevel integrity**; Data integrity is built into the model at the table level to ensure that records are not duplicated and to detect missing Primary key values; at the relationship level to ensure that a relationship between a pair of tables is valid.

- **Logical and physical data independence from database applications**; changes made by a user to the logical and physical design of the RDM will not adversely affect the applications built upon it.

- **Guaranteed data consistency and accuracy**; data is consistent and accurate due to the various levels of integrity which can be imposed within the database.

- **Easy data retrieval**; at the user's command data can be retrieved either from a particular table or from any number of related tables within the database.

A Relational Database Management System, or RDBMS, is a software program that is used to create, maintain, and manipulate a relational database. A RDBMS is also used to create applications that users can interact with the data stored in the database.

As the benefits of the RDBM became more widely known, many companies decided to move from hierarchical and network database models to the relational

database model, thus creating a need for more and better mainframe RDBMS programs. The 1980's saw the development of various commercial RDBMS's for mainframe computers such as Oracle, and IBM's DB2.

## C. OBJECT ORIENTED AND OBJECT RELATIONAL DATABASE MODEL

Object Oriented Programming (OOP), a new style of programming, began to be used in the late 1980's. OOP has shown significant advantages over traditional programming. These advantages will be explained in the following paragraphs. This change in programming also brought a new understanding for database models, called Object Oriented DBMS (OODBMS).

An OODBMS is basically a DBMS that integrates the database capabilities with object-oriented capabilities. An OODBMS makes database objects appear as programming language objects. Lets look closely at how it works:

Objects are software representations of real world entities. To capture the features and capabilities of the real world, objects consist of both attributes and operational characteristics. Classes are the templates for objects, which are similar to one another in behaviors and attributes. One of the important features of the Object Oriented Model (OOM) is encapsulation. Encapsulation is the access to the data that is stored in the object through well-defined behaviors or interfaces approved or accepted by that object. The capability of an object-oriented database to add new objects and their associated behaviors without affecting other objects (entities in database level) is very flexible. In the OOM, new objects can be defined in terms of existing objects that are known as base classes. Polymorphism allows several versions of the same behavior to exist in the subclasses, but the proper version of this behavior is invoked at runtime depending on the class of the object. (Kroenke David M., 1998, p. 22)

What are the advantages of OODBMS over RDBMS?

The first advantage of OODBMS is that OODBMS is integrated with object oriented programming languages. The programmer need not learn a programming language and then learn SQL. Using the language will automatically provide object persistence. In theory, a programmer can code an instruction that causes object methods to be invoked, and the OODBMS will find the appropriate methods, load them in memory, and cause them to be executed.

Second, the OODBMS provides for the definition of user-defined types. Unlike traditional DBMBS products where the basic data types are hard-coded in the DBMS and are unchangeable by the users, with an ODBMS the user can encode any type of structure that is necessary and OODBMS will manage that type. (Kroenke David M., 1998, p. 335)

Because of user defined types, and because relationships are defined in context, it is easy to define complex data in an ODBMS. Unlike the RDBMS, there is no need to define 1:1 or 1:N or N:M relationship and create the appropriate foreign key structures. Instead, the programmer defines the relationship in context, and the OODBMS creates the necessary data structures in the database.

Finally, the OODBMS automatically creates persistent object ID's. This not only saves programmers work, but it also enables the OODBMS to provide single-level memory so that the programmer need not be concerned with whether or not an object is located in memory. (Kroenke David M., 1998, pp. 341-342)

Oracle, being a RDBMS vendor, did not want to be left out of OODBMS, so Oracle added another terminology to the DBMS history, Object Relational Database

Management Systems (ORDBMS). Oracle claims that ORDBMS will dominate over object oriented technology, because object relational databases have the power of both the RDBMS and OODBMS. The Object Relational approach allows organizations to use their pure relational database and still get the benefits of object-oriented features.

As a result, a new type of DBMS-the object oriented DBMS-has been developed. While OODBMS have significant advantages over traditional techniques for object storage, most data in commercial systems today are in relational format. Organizations are reluctant to spend the time and money necessary to convert all of their files and relational data to the OODBMS format. Furthermore, current OODBMS's lack some features that are essential in the commercial environment. Hence, at present, OODBMS are not seeing widespread use. However, this may change in the near future. (Kroenke David M, 1998, pp. 481-482)

## D. DATA MODELING

Data modeling is the process of creating representation of the developers' view of the data and is the most important task in the development of effective database applications. If the data model incorrectly represents the view of data, the developers will find the applications difficult to develop, incomplete, and very frustrating. Therefore, it is the basis for all subsequent work in the development of databases and their applications.

There are currently two main data modeling approaches; the Entity Relationship (E-R) Model and Semantic Object Model.

### 1. Entity Relationship Model

This model includes four basic elements:

- Entities

- Attributes
- Identifiers
- Relationships

An entity is something that can be defined and in order to track in a database, such as a person, a student or a vehicle. Attributes are the properties of the entities, such as the age of a person or the name of a student.

Identifiers are the attributes that uniquely identify the specific entities. For example, the identifier of the person entity might be a social security number.

A relationship is an association among two or more entities. For example, relationship can be defined as that Mike (Employee entity) works in the Billing Department (Department entity) as shown in the Figure 2.3

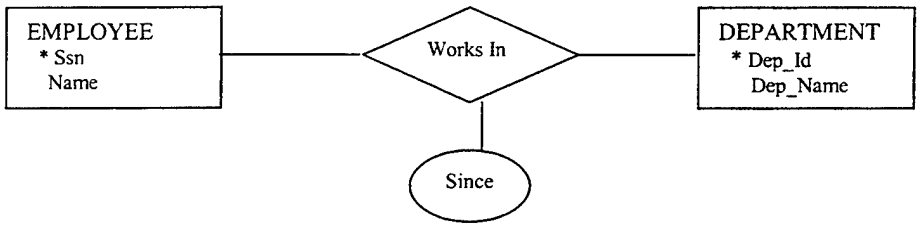



Figure 2.3. The Works In Relationship Set

Relationships between entities may occur in different ways, such as One-to-One (1:1), One-to-Many (1:N), Many-to-One (N:1), or Many-to-Many (N:M). The cardinality ratio for the relationship specifies the number of relationship instances that an entity can participate in. For example, in the "Works In" relationship DEPARTMENT:

EMPLOYEE is of cardinality ratio 1:N meaning that each department can be related to numerous employees, but an employee can be related to only one department.

### 2. Semantic Object Model

This model uses semantic objects instead of entities. A semantic object is a representation of some identifiable thing in the user environment. More formally, a semantic object is a named collection of attributes that sufficiently describes a distinct identity (like entity). The word "sufficiently " is important, because an object must have enough attributes to be a semantic object. (Kroenke David M., 1998, p. 80)

The primary difference between an E-R model and a semantic object model is that entities are the basic focus area in an E-R model, where semantic objects are the focus in the other model.

Figure 2.4 is a simple presentation of a semantic object called DEPARTEMNT.

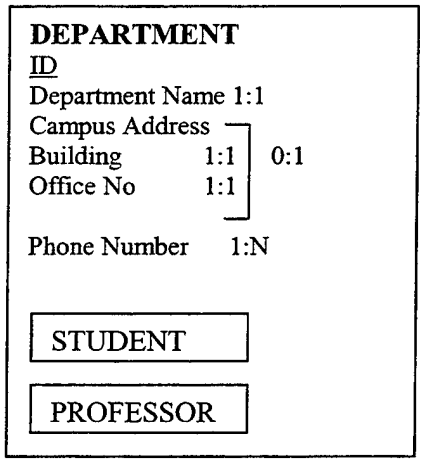



Figure 2.4 A Semantic Model

There are different kinds of objects to represent different kind of data. The basic object types are Simple Objects, Composite Objects, Compound Objects, and Hybrid Objects.

## 3. High Level Conceptual Data Modeling for Database Design

Conceptual modeling is an important phase in designing a successful database application. In this section, I will represent the traditional approach of concentrating on the database structures and constraints during the database design. I will present the modeling concept of the Entity Relationship (E-R) model, which is a popular high-level conceptual data model. Figure 2.5 shows a simplified description of the database design process.

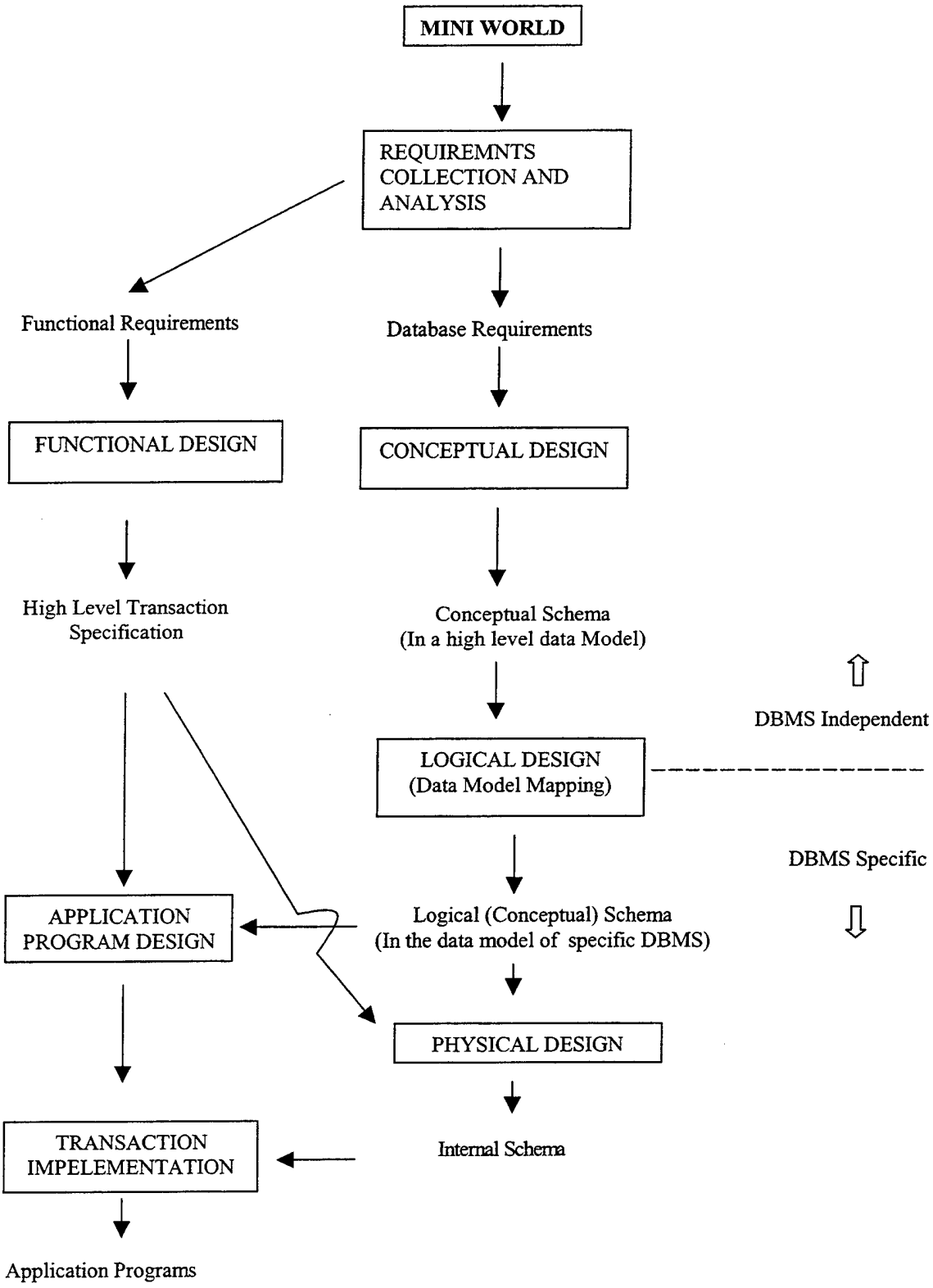



Figure 2.5 Conceptual Design Workflow

The first step shown is **requirements collection and analysis.** During this step, the database designers interview prospective database users to understand and document their **data requirements.** The result of this step is a concisely written set of user requirements. These requirements should be specified in a form that is detailed and complete as possible. In parallel with specifying the data requirements, it is useful to specify the known **functional requirements** of the application. These consist of the user-defined **operations** (or **transactions**) that will be applied to the database, and they include both retrievals and updates. In software design, it is common to use data flow diagrams, sequence diagrams, or scenarios for specifying functional requirements.

Once all of the requirements have been collected and analyzed, the next step is to create a **conceptual schema** for the database, using a high-level conceptual data model. The conceptual schema is a concise description of the database requirements of the user, and includes detailed descriptions of the entity types, relationships, and constraints. These are expressed using the concepts provided by the high-level data model, they are usually easier to understand, and can be used to communicate with nontechnical users. (Elmasri, R. and Navathe S., 2000, p. 43)

The next step in database design is the actual implementation of the database, using a commercial DBMS. Most current commercial DBMS use an implementation of the data model (such as the relational model) so the conceptual schema is transformed from the high-level data model into the implementation data model. This step is called **logical design** or **data model mapping.** (Elmasri, R. and Navathe S., 2000, p. 44)

Finally, the last step is the physical design phase. During this step, the internal storage structures, access paths, and file organizations for the database files are specified.

In parallel with these activities, application programs are designed and implemented as database transactions corresponding to the high-level transaction specifications.

## III. COMPONENENTS AND ARCHITECTURE OF A WEB DATABASE

There are many technologies available to implement a Web database application. Everything is still evolving, but a few reliable categories of software and architectures seem to be emerging. Selecting the best strategy for a Web database application depends on user perspectives, styles, and priorities. In this chapter, I will explain the architectures and describe how Web database components are different from each other and how programmers can use them to solve different kinds application problems.

Before beginning the detailed definitions, let us examine a mechanism, which will aid in understanding the definitions better:

Suppose that you want to buy a book from an on-line retailer and you have typed the URL "http: www.amazon.com" on the address location of your browser. Typing that web page address is actually a way of making a request. One of the main functions of the browser is to make requests on the behalf of users. The web server that runs at the company building is designated to serve our request, which is one of its main functions.

The first page that we see is the homepage of the company, which is usually default.htm or index.htm. This is usually the page where the companies offer their various services such as buying products, making a search or making a query via links. In our case, we will buy a book, so we click on the relevant link, which takes us to another page where we can choose the book we want. Before giving the order to buy, we need to give some personal information for shipment and payment. The most common way to provide the information is by using forms. By the time we click on the "submit" button, we are specifying script parameters to be transmitted via our browser. This is another request made by our browser and sent via the Internet to the web server of the

company. The server running at the company building is, itself, unable to take that information and store it. So, the web server asks for assistance from one of the programs (under one of its directories, i.e., "\cgi\bin\example.exe"), which executes the script. The script communicates with a related database and stores the information in that database's tables. Then, another script is used to return the output to the web server, which is, in our case, simply confirmation information. The web server sends this script output back to the browser, and the browser parses and processes the information.

## A. THE TWO-TIERED CLIENT/SERVER WEB DATABASE

In a two-tier (client/sever) architecture, the computing client talks directly to a server with no intervening process (Figure 3.1). The two-tiered client/server architecture is the most common architecture on microcomputer-based LANs. Hence, the clients manage the user interface, validate data entered by the user, post requests from clients, execute database retrievals and updates, manage data integrity, and control transactions.

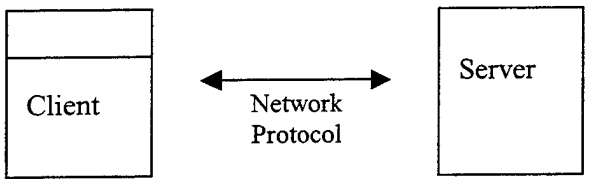



Figure 3.1 Two-Tiered Client/Server Architecture

Web tools and databases are technologies that were developed separately, however both technologies are based on two-tiered client/server architecture (Figure 3.2). The partitioning of functions between a Web browser (client) and a Web server (server) is very distinctive. The Web server delivers HTML pages and the Web browser displays those pages by interpreting the HTML tags. Neither side can change this division of

functions. Because of this simplicity and standardization, many vendors can create web browsers.

When it comes to the partitioning of the functions between the database client and the database server, it is much less distinctive. Decisions about partitioning the functions are often made by application programmers, and are influenced by the requirements of the project. Therefore, there is no standardization. This lack of standardization means that a significant programming effort is usually needed to implement changes to a database client and a database server.

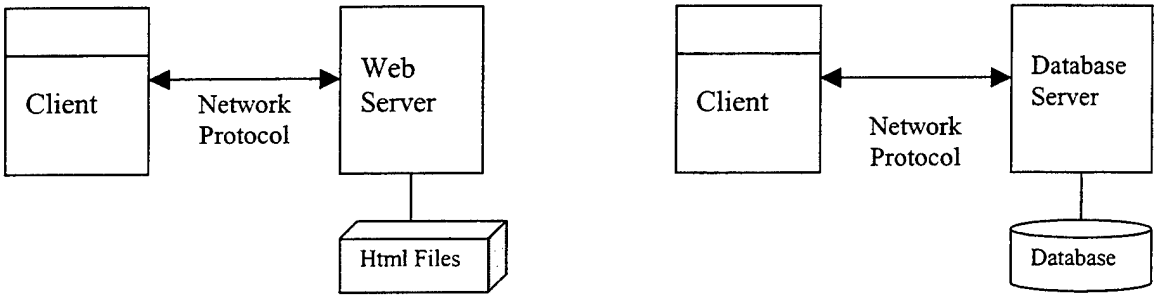



Figure 3.2 Two-tiered Client/Server Web and Database Architecture

Any computer can be a client or a server. The client and the server are often a microcomputer because of cost issues. Sometimes, the server is a mainframe either because of organizational reasons, or when considerable power is required from the server. The clients and servers are generally connected to each other via a LAN.

The typical client-server architecture that is shown in Figure 3.2 works well in relatively homogeneous environments with fairly static rules. For dispersed, heterogeneous environments with rapidly changing rules, there is another client-server architecture, called three-tier client-server architecture. In this type, an additional middle

tier functionality server is added to the configuration (Elmasri, R. and Navathe S., 2000, p. 789)

The main advantage of a client-server database system is that, since the bulk of the database processing is done on the back-end, the speed of the DBMS is not tied to the speed of the client workstation. Because the client is separated from the server, users are no longer limited to one type system platform. The clients can be IBM compatible PCs, Macintoshes, UNIX workstations or any combination of these, and can run multiple operating systems. However, According to the Standish Group, 30% of the client-server projects fail.

## B. THE THREE-TIERED WEB DATABASE

The three-tiered client/server architecture introduces a third layer of processing between the client and the server. Three-tiered architecture is the more recent architecture on PC-based LAN's.

An important advantage of this architecture over two-tiered architecture is that it helps clients and servers to process their works. In other words, it allows clients and servers to lose weight and become "thin clients" and "thin servers". This means that the partitioning of functions can be carried further, and greater modularity can be achieved. It is usually agreed that transactions should be implemented in the middle tier. Other processes that could be implemented in that layer are translating data from legacy applications on mainframes, handling security and authentication, and generating reports.

Web database applications combine their two-tiered parent technologies into a new kind of system. This new system is based on the three-tiered client/server architecture. A web browser occupies the client tier, a database server occupies the server tier, and a middle tier holds a Web server and a server extension program

(Figure3.4). Eventually, this architecture reduces the network traffic, makes components interchangeable, and increases security. However, this architecture also makes database transaction processing more difficult because of the stateless nature of the HTTP protocol that is used to transfer data between the web browser and the database

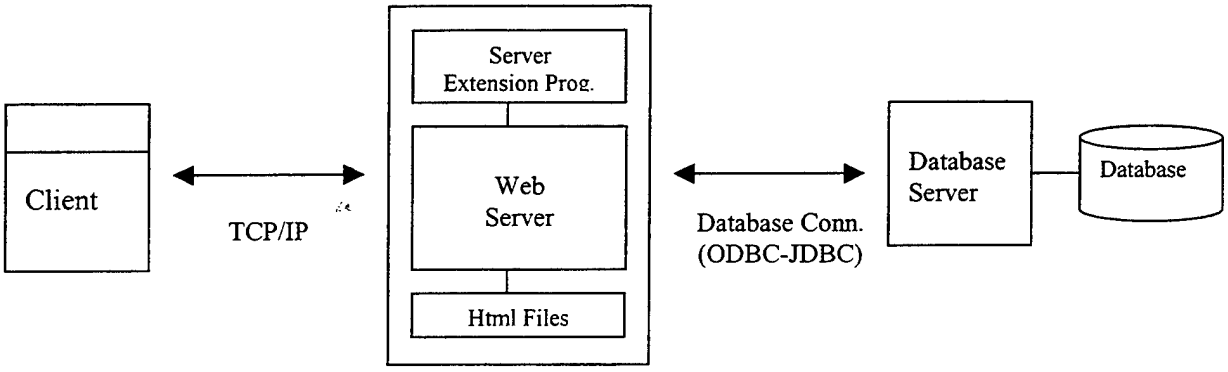



Figure 3.4 Three-Tiered Web Database Application

The Web browser (first tier) sends a Web page request or data request to the Web server. The Web server (second tier) takes the page request and ships the data request to the server extension program. Then, the server extension program accepts the requests and converts them to a form that the database server (third tier) can interpret. For the next step, the database server performs a task, such as query, insert or update, and returns a result set to the server extension program. The server extension program converts the database result to a form that the Web browser can accept (e.g.: HTML), and finally it passes the result set to the Web server, which passes the final result to the Web browser.

## C. SERVER EXTENSION PROGRAMS

One of the most important reasons for using a server extension program in the middle tier is to take advantage of the standards that already exist in the two last tiers by translating between the Web server and the database server. Other reasons for utilizing server extensions include handling database connections to reduce network traffic, and maintaining a pool of open database connections to reduce the overhead associated with opening and closing the database. Server extensions also support interchangeability at their standard interfaces. Thus, Web servers and database servers can be replaced or upgraded with relative ease.

### 1. CGI for Server Extension Program

The Common Gateway Interface (CGI) is a standard way of interfacing external applications with Web servers. The CGI is a mechanism that allows a Web server to run a program or script on the server and send the output to a Web browser. It is important to understand that the CGI is neither a programming language nor a script. It is, rather, the mechanism to enable scripts to operate within standards.

The database server can be made to interact with the Web server via the CGI. So, the CGI scripts, which are written in languages like PERL, C, and Tcl, serves as a middleware. A CGI external application executes in real-time and dynamically produces output information. It processes HTTP requests from Web clients and returns an HTML document.

The main disadvantage of this approach is that for each client request, the Web server must start a new CGI process. Each process makes a new connection with the DBMS and the Web server must wait until the results are delivered to it.

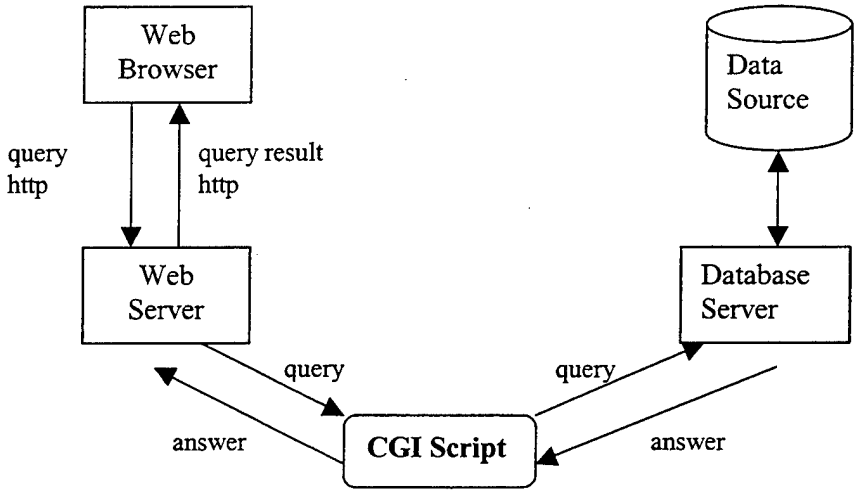



Figure 3.6 Database access on the Web using CGI Scripts

### 2. ISAPI & IDC for Server Extension Program

ISAPI (Internet Server Application Programming Interface) server extensions provide an alternative to the use of Common Gateway Interface applications for Internet servers. Unlike CGI applications, ISA's run in the same address space as the HTTP server and have access to all the resources available to the HTTP server. ISA's have lower overhead than CGI applications because they do not require the creation of additional processes, and they do not perform time-consuming communications across process boundaries. Both extension and filter DLL's may be unloaded if the memory is needed by another process.

An Internet client calls an ISA through the HTTP server the same way it would call a CGI application. For example, a client might call a CGI application as:

`http://sample/example.exe?Param1&Param2`

It would call an ISA that performs the same function as:

http://sample/example.dll?Param1&Param2

The Internet Database Connector (IDC) is an ISAPI application. The IDC enables Web pages to link to databases supporting Open Database Connectivity. Web page developers create an IDC file that resides on the Internet Server. The IDC file is a text file that specifies an ODBC data source name, login information, and a SQL query to retrieve data. Web page developers also create an HTX file, which is a HTML formatting template for the data retrieved from the ODBC-compliant database. The IDC file references the HTX file.

When an IDC file is requested the following sequence of steps occurs:

- The web server recognizes the IDC file is requested, so it passes along the IDC file to the IDC dynamic library (DLL).

- The IDC DLL reads the IDC file, passing the SQL statement and the name of the data source to the ODBC Administrator program.

- The ODBC Administrator program passes the SQL statement to the appropriate ODBC driver, which then passes the SQL statement to the database.

- The database runs the query and returns any rows to the IDC DLL via ODBC.

- The IDC DLL merges the returned rows with the HTX file, producing the standard HTML file, which it passes along the Web Server.

- Finally, the Web Server passes the HTML file back to the browser over the Internet or Intranet connection.

Figure 3.7 represents an example of an ISAPI server extension program for an Internet Information Server (Web Server) and a SQL server ( Database Server):

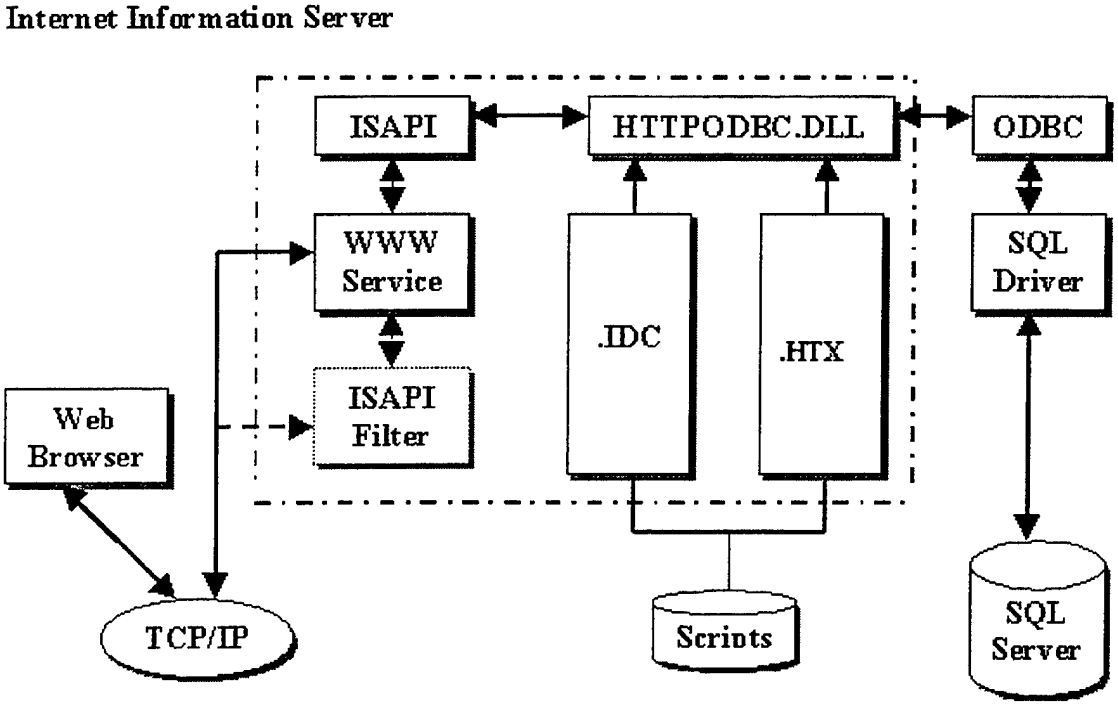



Figure 3.7 IDC to Access the Internet Information Server Log File, Stored in the SQL Server Database

### 3. Remote Method Invocation (RMI) Extension Program

RMI enables the programmer to create distributed Java-to-Java applications, in which the methods of remote Java objects can be invoked from another JVM (Java Virtual Machine), possibly on different hosts. A Java program can make a call on a remote object once it obtains a reference to the remote object, either by looking up the

remote object in the bootstrap-naming service provided by RMI, or by receiving the reference as an argument or a return value. A client can call a remote object in a server, and that server can be a client of another remote object. (JavaSoft Web Site , 1999)

### 4. WebBase for Server Extension Program

WebBase relies on special tags and SQL, but it is tightly integrated with a Web server specially tuned for the database. WebBase supports the standard for Dynamic HTML and Document Object Model. Dynamic HTML lets authors make interactive Web pages that use up far less bandwidth than conventional HTML pages. Dynamic HTML can dynamically modify HTML tags, style sheets, text, tables, ActiveX objects, and Java applets without server intervention. WebBase automatically creates the database and forms to add, update, search and view records in any Browser. Users need no special viewers or plug-ins to use this add-on. (WebBase Web Site, 1998)

In addition, WebBase provides solutions from single access to real estate listings, product pricing, availability, and customer order status to complex catalog ordering applications. Users can make anything available in the database to anyone browsing the Web site, or allow access to a specific audience that users control through password protection. Users can also make existing databases far more powerful by adding hypertext links to reports. This feature allows users to delve into a report in detail, while maintaining the simplicity of high-level view.

### 5. Active Server Pages (ASP)

The Active Server Pages specification is a technology built on top of the Microsoft Internet Information Server (IIS). Unlike the IDC specification that I explained above, ASP is not limited to database connectivity.

An ASP is an HTML page that includes one or more scripts (embedded programs) that are processed on a Microsoft Web server before the page is sent to the client. An ASP is somewhat similar to the Common Gateway Interface (CGI) application, in that both involve programs that run on the server, usually tailoring a page for the user. Typically, the script in the web page at the server uses the input received as the result of the client's request for the page to access data from a database and then builds or customizes the page on the fly before sending it to the requestor. (Buser D. and others, 1999, p. 79)

Since the server-side script is just building a regular HTML page, it can be delivered to any browser. An ASP file is created by including a script written in VBScript, JavaScript or PerlScript (Perl interpreter for Win 32 must be installed) in an HTML file and then renaming it with the ".asp" file suffix.

The main disadvantage of the ASP technology is that ASP does not work in other web servers; it only works with Microsoft Web Server (IIS).

### 6. Servlets and JSP

Servlets are Java technology's answer to Common Gateway Interface programming. They are programs that run on a Web server, acting as a middle layer between a request coming from a Web browser or another HTTP, and databases or applications on the HTTP server. A Servlet is a Java class, and thus needs to be executed by a Java VM, called a Servlet engine. Servlets are loaded by the engine when they are called, and remain running until the servlet is explicity unloaded or the engine is shut down. Their job is to:

- **Read any data sent by the user:** This data is usually entered in a form on a Web page, but could also come from a Java Applet or a custom HTTP client program.

- **Look up any other information about the request that is embedded in the HTTP request:** This information includes details about browser capabilities, cookies, the host name of the requesting client, and so forth.

- **Generate the results :** This process may require talking to a database, executing an RMI or CORBA call, invoking a legacy application, or computing the response directly.

- **Set the appropriate HTTP response parameters :** This means telling the browser what type of document is being returned (e.g. HTML), seeting cookies and caching parameters.

- **Send document back to client:** This document may be sent in text format (HTML), binary format (GIF images) or even in a compressed format like gzip.

JSP is an extension of the Servlets technology. Anything that is done in JSP can be done with Servlets. However, JSP allows to mix static HTML with the code.

Typically, it is also easier to read the code and visualize the page that will ultimately be generated. For instance:

**JSP**

```
<HTML>
<HEAD>
<TITLE>Hello World!!</TITLE>
</HEAD>
<BODY>
Hello World! Your name is:<%
out.println(response.getParameter("name"));
%>
</BODY>
</HTML>
```

**Servlet**

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

public class HelloWorld extends HttpServlet {

    public void doGet(HttpServletRequest request,
    HttpServletResponse response) throws IOException,
    ServletException {

        response.setContentType("text/html");
        PrintWriter out =
        response.getWriter();

        out.println("<HTML>");
        out.println("<HEAD>");
        out.println("<TITLE>Hello
        World!!</TITLE>");
        out.println("</HEAD>");
        out.println("<BODY>");
        out.println("Hello World! Your
        name is: "
        +
        response.getParameter("name"))
        ;
        out.println("</BODY>");
        out.println("</HTML>");

    }

}
```

In a nutshell, the JSP page is being converted to a normal servlet, with static data being written to an output stream. There are ways to reduce the actual amount of code written in the Servlet. Even though they both generate the same output, JSP is easier to read and easier to write. Clearly, JSP and Servlets have their own distinct roles and uses which allow the developer freedom and ease of use.

Some advantages of Servlet and JSP over CGI programming are as follows: (Hall M, *Core Servlets and JavaServer Pages*, p. 11, 2000)

- **Efficiency** - Using CGI programming, each time an HTTP request is received a new process is started, which can result in poor performance and scalability issues. Using Servlets, the Java VM is always running, therefore starting a Servlet creates a Java thread, as opposed to a system process.

- **Power** - Servlets allow power unknown by traditional CGI. They allow programmers to do things that would either be very difficult or otherwise impossible because they have access to the entire family of Java APIs. Servlets easily share data and maintain information, making session tracking.

- **Security** - Servlets can by run by the Servlet engine in a restrictive sandbox, similar to a web browser's sandbox for applets. The helps to protect against malicious Servlets.

- **Portability** - The Servlet API takes advantage of the Java platform. It is a fairly simple API which is supported by nearly all web servers so that Servlets may be moved from platform to platform, usually without any modification whatsoever.

## B. DATABASE CONNECTION PROGRAMS

### 1. Open Database Connectivity (ODBC)

The Open Database Connectivity (ODBC) interface is a C programming language interface that makes it possible for applications to access data from a variety of database

management systems (DBMS). The ODBC interface permits maximum interoperability; an application can access data in diverse DBMS through a single interface. Furthermore, that application will be independent of any DBMS from which it accesses data. Users of the application can add software components called drivers, which interface between an application and a specific DBMS. The use of drivers isolates applications from database-specific calls in the same way that printer drivers isolate word processing programs from printer-specific commands. Because drivers are loaded at run time, a user only has to add a new driver to access a new DBMS; it is not necessary to recompile or relink the application.

The server extension program translates Web browser requests into an ODBC SQL statement, submits them to the data source via the ODBC driver, and retrieves the results. An example of an ODBC function call is SQLConnect, which connects to the data source when given a data source name, a user ID and password, and a few other parameters.

The ODBC architecture has four components: ( Microsoft Web Site, 200)

- **Application:**

  Performs processing and calls ODBC functions to submit SQL statements and retrieve results.

- **Driver Manager:**

  Loads and unloads drivers on behalf of an application; processes ODBC function calls or passes them to a driver.

- **Driver:**

Processes ODBC function calls, submits SQL requests to a specific data source, and returns results to the application. If necessary, the driver modifies an application's request so that the request conforms to syntax supported by the associated DBMS.

- **Data source:**

Consists of the data the user wants to access and its associated operating system, DBMS, and network platform (if any) used to access the DBMS.

The following illustration shows the relationship between these four components:

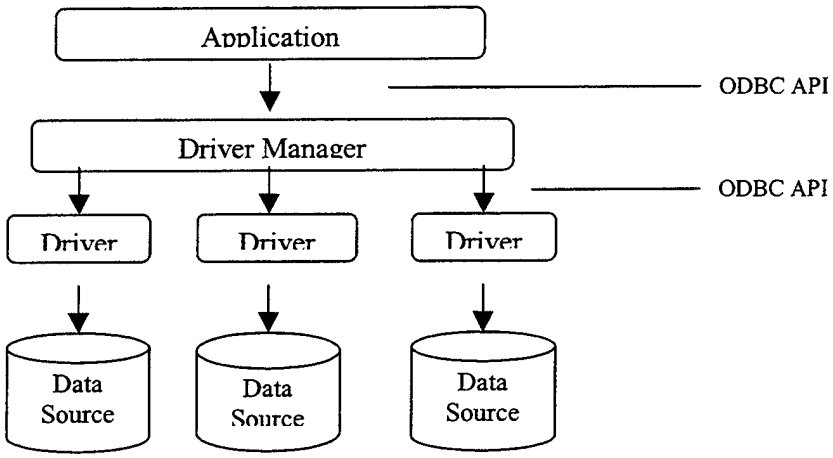



Figure 3.5 Relationship between ODBC components

As seen from Figure 3.5; multiple drivers and data sources can exist, which allows the application to simultaneously access data from more than one data source. Second, the ODBC API is used in two places: between the application and the Driver

Manager, and between the Driver Manager and each driver. The interface between the Driver Manager and the drivers is sometimes referred to as the *service provider interface,* or *SPI.* For ODBC, the application programming interface (API) and the service provider interface (SPI) are the same; that is, the Driver Manager and each driver have the same interface to the same functions. (Microsoft Web Site, 200.)

### 2. Java Database Connectivity (JDBC)

JDBC (Java Database Connectivity) is a Java API (Application Programming Interface) for executing SQL statements. More precisely; JDBC is a collection of Java classes and interfaces that enables database access from programs written in the Java programming language. The classes and interfaces are part of the "java.sql " package.

Using JDBC, it is easy to send SQL statements to virtually any relational database. In other words, with the JDBC API, it is not necessary to write one program to access a Sybase database, another program to access an Oracle database, and so on. One can write a single program using the JDBC API, and the program will be able to send SQL statements to the appropriate database. And, with an application written in the Java programming language, one also does not have to worry about writing different applications to run on different platforms. The combination of Java and JDBC lets a programmer write it once and run it anywhere.

There are three basic types of JDBC drivers. The first one is simply a bridge between the JDBC API and the operating system-dependent ODBC driver manager. Sun provides these bridges for OS/2,Win32 and Solaris systems. These kinds of drivers were created to allow quick development of applications without waiting for database suppliers to create real JDBC drivers.

The second driver uses JNMI (Java Native Method Invocation) to call the functions of the database client library. These are also platform-dependent, as the database supplier will have to provide both the client libraries and the JDBC driver specific for the desired platform. IBM and Oracle, among others, provide this kind of drivers

The third driver is a "100% Pure" Java driver. It may not be certified as such, but the important feature is that the driver is written entirely in standard Java, and then it can be used on any platform that has a Java Virtual Machine. This is perfect if the database supplier does not support the client platform. For example, Linux, Oracle, Sybase and BM, provide this kind of driver; and some can be obtained from third parties or as Open Source Software. (Lozano F., 2000)

### 3. JDBC versus ODBC

Microsoft's ODBC API is probably the most widely used programming Interface for accessing relational databases. It offers the ability to connect to almost all databases on almost all platforms. So why not just use ODBC from Java?

The answer is that one can use ODBC from Java. In that case, the question now becomes "Why does one need JDBC?" There are several answers to this question: (Olivia R., 1998)

- A Java API like JDBC is needed in order to enable a "pure Java" solution. When ODBC is used, the ODBC driver manager and drivers must be manually installed on every client machine. Since the JDBC driver is written in Java language, JDBC code is automatically installable, portable, and secure on all Java platforms from network computers to mainframes.

- ODBC is not appropriate for direct use from Java because it uses a C interface. Calls from Java to native C code have a number of drawbacks in the security, implementation, robustness, and automatic portability of applications.

- ODBC is hard to learn. It mixes simple and advanced features together, and it has complex options even for simple queries. JDBC, on the other hand, was designed to keep simple things simple while allowing more advanced capabilities where required.

- A literal translation of the ODBC API into a Java API would not be desirable.

- ODBC drivers cannot connect to remote databases, while JDBC drivers can easily make a connection to remote databases.

## C. CLIENT-SIDE EXTENSION PROGRAMS

The HTML passed back to a web browser by a web server may contain more than the HTML. For example, a client-side ActiveX control may play background music or a Java applet may create special effects. So, a client-side extension is a program that adds to the capabilities of a web browser. Client-side extensions may be used for many purposes, but one of their main functions is to perform input field validations. Although

there are no formal classifications for client-side extensions, they fall into three categories: pluggable applications, Java applets, and scripts.

Pluggable applications are stand-alone programs that run on the Web browser; their purpose is to process and display data that the browser cannot handle directly. (i.e., ActiveX)

Java Applets are compiled programs that are downloaded when a HTML page is requested and are then run by the browser. Applets run as byte-code interpreted programs, which reduces the likelihood that they will transmit a virus, since each instruction is validated before being run.

Scripts are programs embedded in a HTML page. Scripts integrate well with the web browser because they add functionality without changing the look and feel of standard web page. (i.e., Dynamic HTML, JavaScript, VBscript)

# IV. ORACLE 8I WEB DATABASE ARCHITECTURES

In this chapter, I will present the Oracle 8i architecture for web information system design. There are two main approaches offered by Oracle: a two-tier architecture with Oracle WebDB, and three-tier architecture using the Oracle Application Server (OAS).

## A. TWO-TIER ARCHITECTURE WITH ORACLE WEBDB

As its name implies, WebDB has something to do with putting a database together with a web site. Oracle WebDB is the fastest and easiest way to "Web-enable" Oracle databases. Oracle WebDB comes with tools necessary to build dynamic Web applications and content-driven Web sites. Users do not need to know any HTML or SQL to use it. More precisely, WebDB is an Oracle development environment for building and monitoring content-driven web sites and data driven applications.

WebDB provides a complete web development environment, which includes an HTML server so it can process HTTP requests and serve web pages. The only client software needed to develop and deploy an application is a web browser. (Netscape Navigator 3.1 or later or Microsoft Internet Explorer 4.0 or later.)

WebDB uses the database's native components. It is a collection of PL/SQL procedures contained entirely within an Oracle database. It also includes a lightweight listener that acts as a Web server and a PL/SQL interface to the database. Since it is essentially written in the database's native language, it eliminates required layers such as ODBC or JDBC. The following figure is the representation of Oracle WebDB architecture. (Bradley D., 1999, p.224)

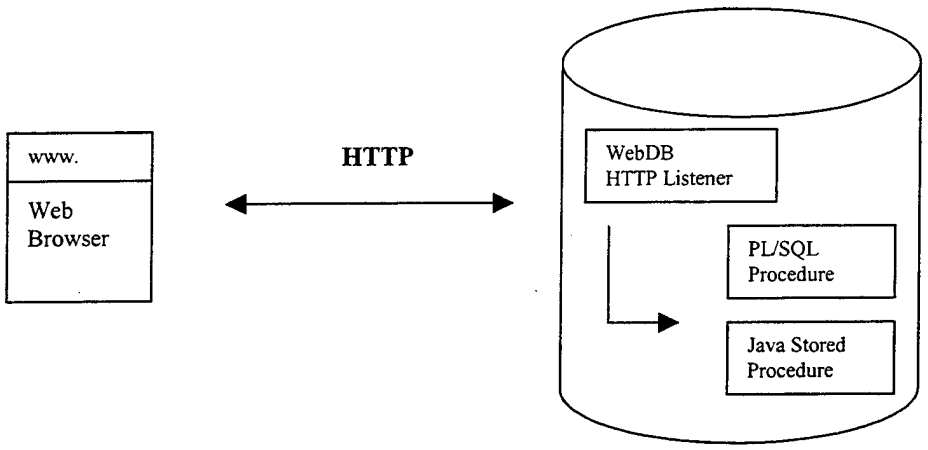



Figure 4.1 Oracle Two-tier Architecture with WebDB

The main disadvantages of Oracle WebDB is that the lightweight listener does not have the ability to handle a large number of requests, nor does it support some important security related technologies, such as Secure Socket Layers (SSL) and IP restriction.

## B. THREE TIER ARCHITECTURE WITH ORACLE APPLICATION SERVER

As mentioned before, in the three-tier approach, there is a middle tier between the clients and the database server. This proxy (middle tier) is called an application server. The application layer is where all the processing is done according to the logic implemented by the system. Business rules are enforced, data integrity is checked, and complex processing dictated by system requirements are carried out. This layer is the workhorse of the three-tier approach.

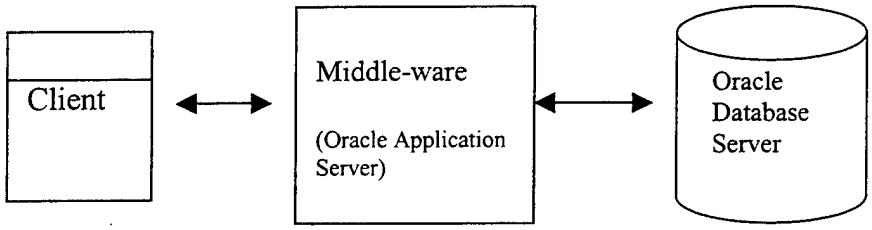



Figure 4.2 Oracle There-tier Architecture

In a three-tier approach the client (sometimes called presentation layer) is where information is displayed to use, and input is accepted for processing. At the middle tier, we can use any web server such as Microsoft Internet Information Server, Apache Web Server or Oracle Application Server, on which I will focus .

The data layer (Database Server, often called back end) plays a role in the storage of information to satisfy requests placed by the other two layers. In many but not all cases, this is a relational database.

This chapter will concentrate on the last two layers; the Oracle Application Server and the connection between OAS and the Oracle Database Server. The first layer was detailed in Chapter III and nothing changes with the client side in Oracle's three-tier approach.

### 1. Oracle Application Server (OAS)-Second Tier

Oracle Application Server (OAS) performs all functions of a standard web server, while keeping a tight integration with a backend Oracle database. OAS is an add-on tool to an Oracle database and is used to build web-based database applications and serve as the main engine between web requests and the Oracle database. More precisely, OAS is an extensible web server that uses plug-in programs called cartridges for database

connections. OAS allows programmers to develop database-integrated systems in a variety of languages, including Java, Perl, and PL/SQL.

OAS supports two different types of dynamic web technology: Common Gateway Interface (CGI) and cartridges.

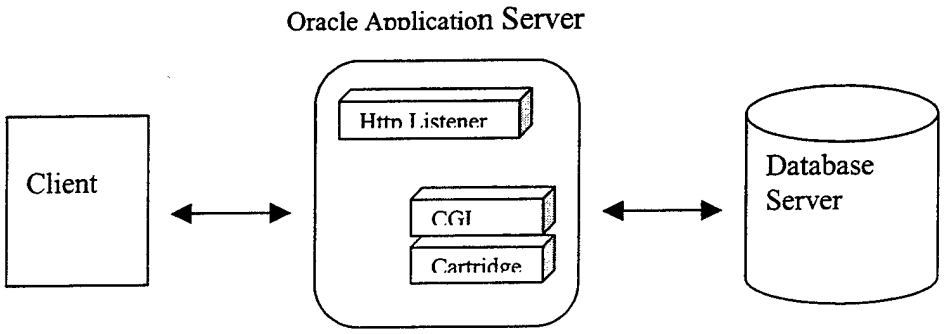



Figure 4.3 Oracle Application Server

CGI allows the programmer to execute any kind of server side program, whether written in a third generation language like C, a scripting language like Perl, or a database language like PL/SQL. There are some important limitations of the CGI interface in OAS.

CGI requires significant overhead. The system must create a process for each CGI interface connection, and allocate resources for it. More over, a CGI program must establish a new connection to the database every time it is executed. This affects performance, especially when a series of CGI executions are linked together to form even a small application.

Another way that OAS takes care of dynamic pages is through cartridges. This method has advantages over CGI. A cartridge is a code module that interacts with the

OAS through a standard interface. The basic function of the OAS is to manage the interaction of cartridges or more accurately, cartridges instances. In short, a cartridge includes code that executes application logic. For example, the Java Cartridge contains code that enables it to connect to Oracle databases and execute Java stored procedures in the database.

The cartridge interface maintains a pool of processes that are already running and connected to the appropriate database. Therefore, using a cartridge is much faster than using a CGI. Unlike CGI programs, cartridge servers do not have to be started for each request. The OAS listener and dispatcher components route incoming request to running cartridge servers, based on the current server load. The cartridge server that handles the request does not need to run on the same machine that initially received the request.

The Oracle Application Server is composed of the following three layers:

- The HTTP Listener Layer
- The Oracle Application Server Layer
- The Application Layer

**The HTTP Listener layer**, handles communication between clients and the Application Server layer through standard Internet protocols.

**The Oracle Application Server layer** manages the creation of cartridge instances, load balancing between multiple instances of individual cartridges, and services to cartridges. This layer is the glue, that holds everything together as well as the location of the basic services such as authentication, logging, failure recovery, transaction control, and load balancing. With Object Request Broker (ORB), it also allows for a

distributed system, where applications, listeners, and data can be located on different physical machines, but still configured and managed from a centralized interface. (William G, 2000, p. 834)

**The application layer**, is the place where specific cartridges (i.e., Java cartridge) are used to implement specific application functionality. Let us examine at the application layer elements (cartridges) in detail.

***a) Java (JWEB) Cartridges***

Java is a modern, object-oriented programming language, and is portable across operating systems and hardware. An amazing number of technologies are arising around Java including, Java Cartridges in OAS, Servlets, EJB, CORBA, RMI, JDBC, SQLJ. These can be used in conjunction with each other. In addition to new technologies, hundreds of classes are built into the class libraries that accompany Java. Additionally, countless classes are available for free or commercially; therefore developers do not need to build all the classes.

Java is currently the premier application development language for the Internet, and as far as the immediate future, this is not likely to change. Java is gaining acceptance and a significant share of the market. Over 70 percent of global 1000 companies are currently either implementing or have specific plans to implement Java solutions to business problems. There is a growing base of Java developers, and a corresponding growth in demand for Java developers. Additionally, there are established standards used in the industry that have proven themselves in production environments. (Bradley D., 2000, p. 801)

The Java Cartridge is generally a source of some confusion. The Java cartridge does not have anything to do with Java applets. A Java applet is a small

program, which is downloaded and run in the browser (client). Essentially, the Java cartridge runs a Java application on the server and returns straight HTML to the browser. The browser needs no special capability to see the application because it is not being asked to do anything but interpret the HTML (the server has already done everything for it). Java Cartridge can be discussed as two different topics: Java cartridges (JWeb) and Servlet cartridges (JServlet). I will explain the Java cartridges first.

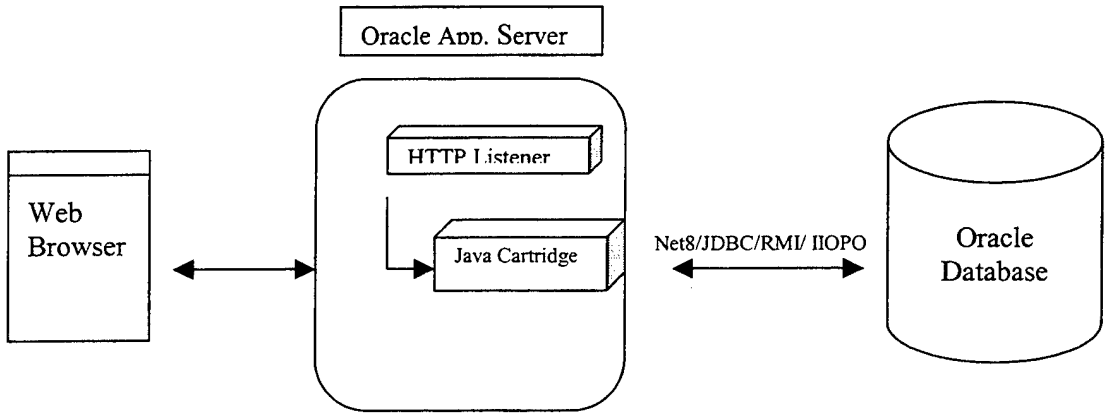



Figure 4.4 OAS Server Java Cartridge

The Java (JWeb) Cartridge comes with the JWeb Toolkit, which is a set of Oracle specific Java classes that developers can use to generate HTML pages and access Oracle databases. These classes help developers create web pages much more easily than the Servlet classes. The Java cartridge also takes care of load balancing, scalability, monitoring, sessions, and other futures of the Oracle Application Server. The Java cartridge minimizes use of the resources by running multiple Java applications on the same virtual machine, as well as handling multiple requests for the same application using the same instance of the application. (William G, 2000, pp. 834-835)

Since the Java cartridge is a runtime environment, it does not have built-in debugging facilities, other than using print statements to generate messages to standard output or to a log file. The Java cartridge does not have a standard interface. JWeb toolkit works within the context of the application server. Therefore, it is better to build and debug the applications outside of the application server and then finish the application using the JWeb toolkit classes.

Now, let us look at how the Servlet cartridge works in OAS:

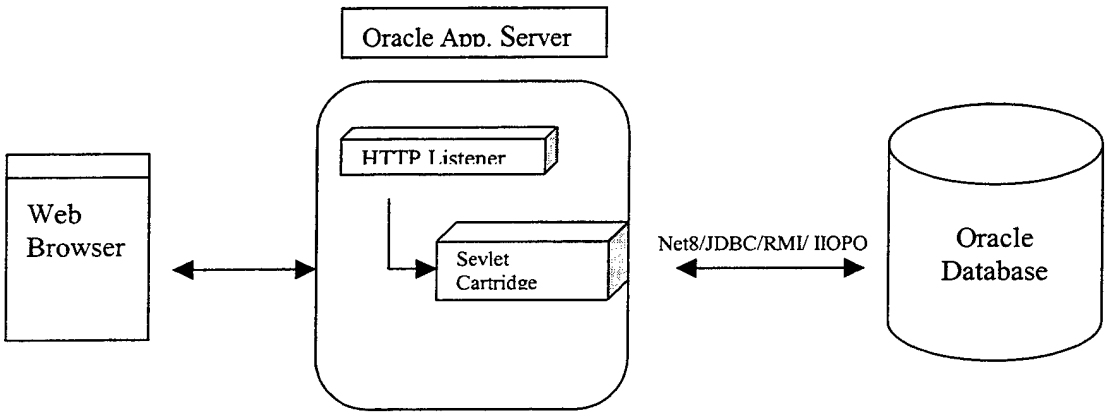



Figure 4.5 Oracle Application Server with Java Servlet Cartridges

The JServlet cartridge contains a Java Virtual Machine and Java class libraries. It provides a runtime environment for server-side Java applications written with the Java Servlet API specification. Servlets are currently the most popular Java technology for building dynamically generated web pages. They are easy to build because all Servlets have the same life cycle characteristic and provide a standard set of method calls. Servlets also support user authentication and threads. A single instance of a servlet can support all client requests for its services.

Like in JWeb, the JServlet cartridge minimizes the use of system resources by running multiple JServlet cartridges on the same virtual machine when they belong to the same application. Free instances of applications are also used when available, instead of creating new instances. The JServlet cartridge comes with the JServlet Toolkit.

In conclusion, for applications that involve complex object-oriented or highly CPU-intensive operations, the Java cartridge or Servlet cartridge are faster than other cartridge methods that I explain in the following paragraphs. Since Java is an object oriented language, Java provides elegant facilities from which to inherit existing Java types and build complex class hierarchies. Furthermore, when natively compiled using the Java compiler (NCOMP), Java's performance will improve significantly.

***b) PL/SQL Cartridges***

With the PL/SQL cartridge, developers can develop Web applications by using Oracle stored procedures. The PL/SQL cartridge provides an environment that enables users to use their browsers to invoke PL/SQL procedure. The stored procedures can retrieve data from tables in the database, and generate HTML pages that include the data to return to the client browser.

The main advantage of the PL/SQL cartridge is that it hides the complexity of interacting with the database. Basically, when it receives a request, the cartridge server logs in to the database, executes a stored procedure, and returns any result to the browser. It is easier to interact with database objects in PL/SQL than in any other language.

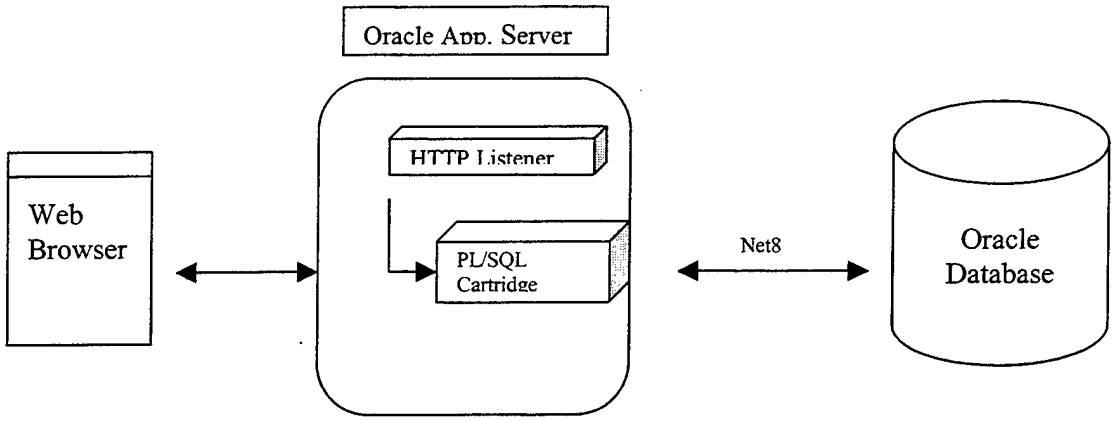



Figure 4.6 PL/SQL Cartridge in OAS

The PL/SQL cartridge connects to the database with the Oracle Net8 communication protocol, and invokes the procedure in the database. The procedure generates the HTML page, which includes the data that is retrieved from the database.

The stored procedure that the cartridge invokes should return the HTML data back to the client. To simplify this task, the PL/SQL cartridge comes with the PL/SQL Web Toolkit, which is a set of packages that you can use in your stored procedure to obtain information about the request, construct HTML tags, and return header information to the client. (Allen C., 1999, p. 10)

Currently, invoking the PL/SQL cartridge is quicker than invoking other cartridges. PL/SQL is a sophisticated procedural language for developing database applications and is ideally suited for building SQL/data-intensive applications. It has information hiding, overloading, and exception handling features. However, PL/SQL does not yet support constructs such as inheritance, polymorphism, and component models that are familiar to distributed system developers.

In conclusion, while PL/SQL is generally more optimized for SQL intensive applications, the overall performance of the application will depend on the relative balance between computational operations and the number of SQL access.

### *c) Perl Cartridges*

Before the advent of applications servers, most CGI scripts were written in Perl. Running Perl as a CGI application has some drawbacks. For one thing, like every CGI script, every time a user makes a request the interpreter needs to be started because Perl is an interpreted language. This can be a serious resource and performance drain.

The Oracle Application Server solves this big performance problem with the Perl Cartridge. Perl cartridge is basically a version of the Perl interpreter that remains resident in memory, waits for requests, and then executes them.

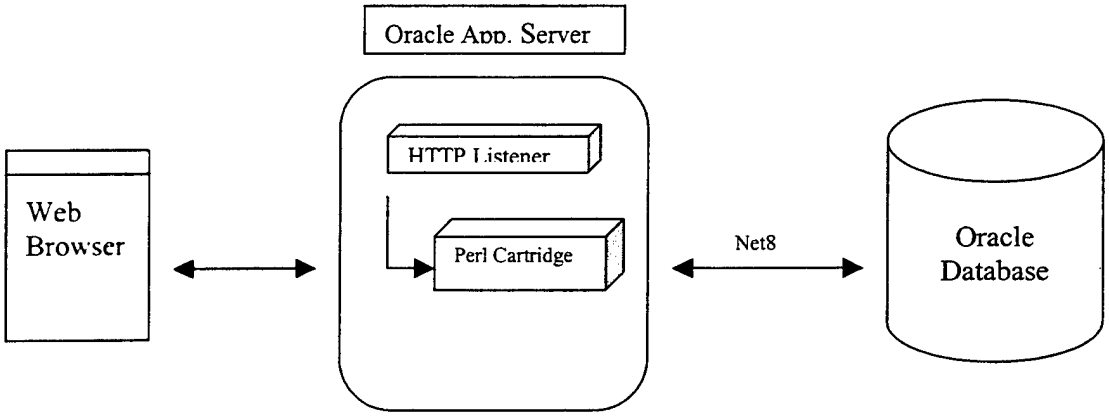



Figure 4.7 Perl Cartridge in OAS

Perl is free and can be used to call operating system commands or programs, whereas PL/SQL cannot. Another major reason for the popularity of Perl is the Apache Web Server that can be deployed in large-scale enterprises supports Perl robustly.

Oracle supports Perl, as Perl has been used since the beginning of web development. Java can be used everywhere, that Perl can be used and Oracle is moving in the direction of Java, rather than Perl.

### *d) LiveHTML Cartridges*

LiveHTML is Oracle's name for Server Side Include (SSI). SSI was one of the web's earliest methods to provide dynamic content in a page. It allows the developer to create a static HTML page, but it embeds dynamic information and sends the result back to the browser. Scripts can be inserted between standard HTML tags. This allows developers to perform more complex commands than those supported by SSI.

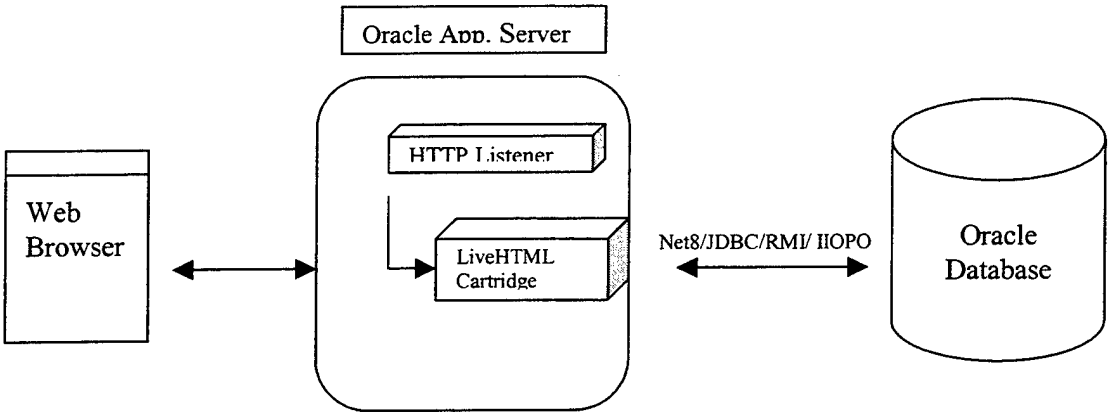



Figure 4.8 LiveHTML Cartridge in OAS

LiveHTML technology is very similar to Microsoft Active Server Pages (ASP) technology. ASP uses Visual Basic as its underlying language, whereas Oracle's LiveHTML uses Perl as its script language.

When developers want to generate an HTML page dynamically, they usually have to write a script or program to generate the entire page, including the static

portions. This requires more time to write the scripts and programs and to generate each dynamic page. LiveHTML provides an alternative method of generating dynamic HTML pages. It saves developers from generating the entire HTML page each time it is requested by allowing them to embed server-side commands and scripts in a static HTML page.

After discussing the cartridges in Oracle Application Server, it is time to mention the connection between OAS and Oracle Database.

### 2. Connection Between OAS and Database server

There are four different ways to connect from OAS to Oracle Database Server. Net8, JDBC, IIOP, and RMI:

#### *a) Net8*

Net8 is Oracle's protocol to connect to remote databases. The main function of Net8 is to establish network sessions and transfer data between a client machine and a server or between servers. Net8 should be installed on both machines to communicate. It is a free utility, which can be downloaded from Oracle's web site.

Another feature of Net8 is the Oracle Security Server. It adds special security features to network traffic of Net8 nodes. The security server uses cryptography across the network and requires users enter a password and login ID. The server keeps track of which users are allowed to access which Oracle8i databases.

Net8 offers network load balancing by allowing configuration of multiple connection routes from clients to a single data source. This configuration feature maximizes performance in the Oracle8i Parallel Server and replicated environments, and provides capabilities from recovering connection failures. (Abbey M. and others, 1999, p. 23)

Moreover, as an extension of Net8 Oracle has its own version of a firewall for protection across the Internet. This software is called Connection Manager. Connection Manager can be configured according to which IP addresses are allowed to access the database.

### *b) Java Database Connectivity (JDBC)*

JDBC is a Java Class library that provides access to relational data. JDBC is an object-oriented application programming interface (API), with interfaces defined by JavaSoft. Conceptually similar to Microsoft's ODBC (Open Database Connectivity), the JDBC API defines Java classes to represent connections, SQL statements, result sets, and other database objects that enable a Java program to interact with a Oracle database. (JDBC was discussed in detail in the previous chapter).

### *c) Internet Inter-ORB Protocol (IIOP)*

IIOP is an object-oriented protocol that makes it possible for distributed programs written in different programming languages to communicate over the Internet. IIOP is a critical part of a strategic industry standard. The Common Object Request Broker Architecture (CORBA) and IIOP are competing with a similar strategy from Microsoft, called the Distributed Component Object Model (DCOM)

CORBA and IIOP assume the client/server model of computing, in which a client program always makes requests and a server program waits to receive requests from clients.

Any CORBA object that is written in any CORBA standard programming language can use IIOP connections. However, because it supports multiple languages, IIOP has some overhead and is slower than other communication protocols.

Oracle uses IIOP to access CORBA objects in an Oracle database.

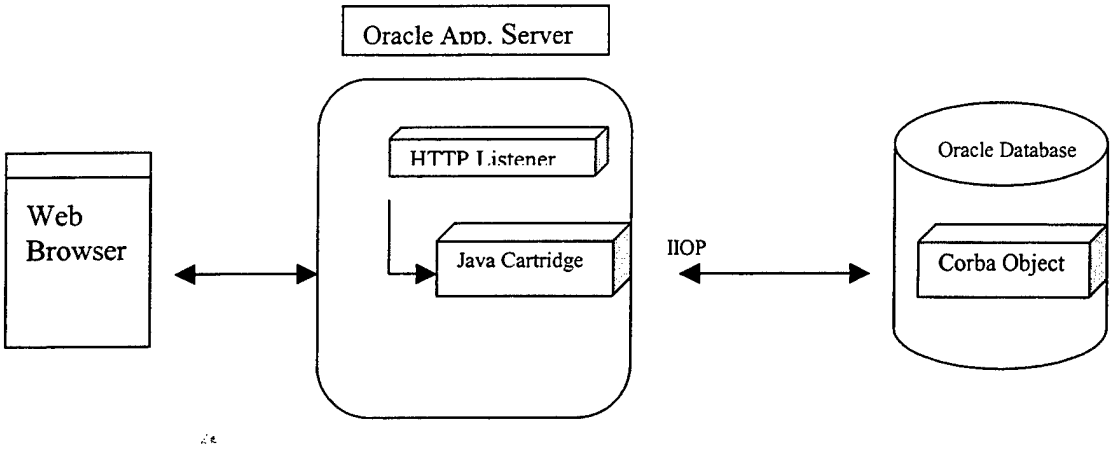



Figure 4.8 IIOP Connection to Oracle Database

### *d)* Remote Method Invocation (RMI)

RMI is a way that a programmer can write object-oriented programs using the Java programming language and development environment, in which objects on different computers can interact on a distributed network.

RMI is the Java version of what is generally known as remote procedure call, but with the ability to pass one or more objects along with the request. The object can include information that will change the service that is performed in the remote computer.

RMI technology is similar to IIOP technology, except RMI supports only Java. RMI originally had less overhead than IIOP since it only supported the Java programming language. However, Oracle uses RMI on top of the IIOP protocol, so by this usage, RMI has as much overhead as IIOP has in Oracle.

Oracle uses RMI to invoke Enterprise Java Beans in Oracle Database server.

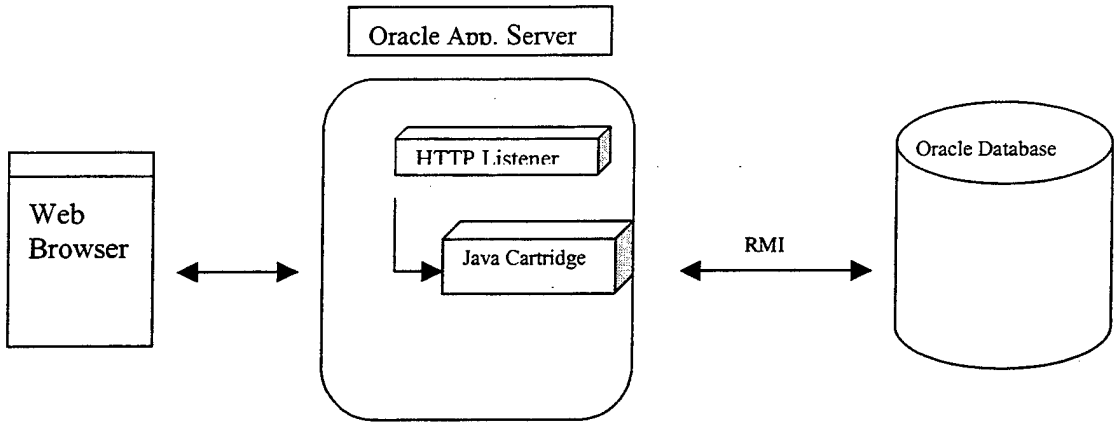



Figure 4.9 RMI Connection to Oracle Database

### 3. Oracle Database Server -Third Tier

The Oracle8i Database Server has the ability to store business logic in many ways. The reason for the widespread use of business logic in the database server procedure is that result set processing improves application performance by eliminating network traffic bottlenecks and allows more efficient use of server resources.

Beginning with the Oracle 8i, application developers started using Java to implement business logic in the database level. The business logic they developed was deployed and stored as program units that run in the database as stored procedure, functions or triggers. Before Oracle8i, PL/SQL was used to implement the business logic on the Oracle Database Server.

Because Java programs (stored procedures) are executed on the database server, SQL access is much faster than when the data must be retrieved from the server to a Java VM on another machine. However, the Java VM still runs slower than PL/SQL on Oracle 8i Database Server.

Java can be used in anywhere that PL/SQL is traditionally used. By adding Java as a server programming language, Oracle aims to open the RDBMS as a general-purpose server platform to all Java developers.

Finally, the two-way interoperability that Oracle provides between Java and PL/SQL allows the reuse of applications. Existing PL/SQL stored procedures can easily be reused from Java. In addition, Java procedures can be reused from PL/SQL.

There are three different ways to implement business logic with Java in the Database Server:

***a) Enterprise Java Beans (EJB)***

EJB are basically distributed Java components that implement a set of predefined Java interfaces. These interfaces are designed for transaction processing functionality. EJB is a powerful development methodology for distributed application development.

EJB enables developers to design and package applications in components that can be assembled with components written by other developers. Component-based programming is very useful because of the prospect of reusable application code, easy assembly of applications by wiring components from different vendors, and flexibility of deployment.

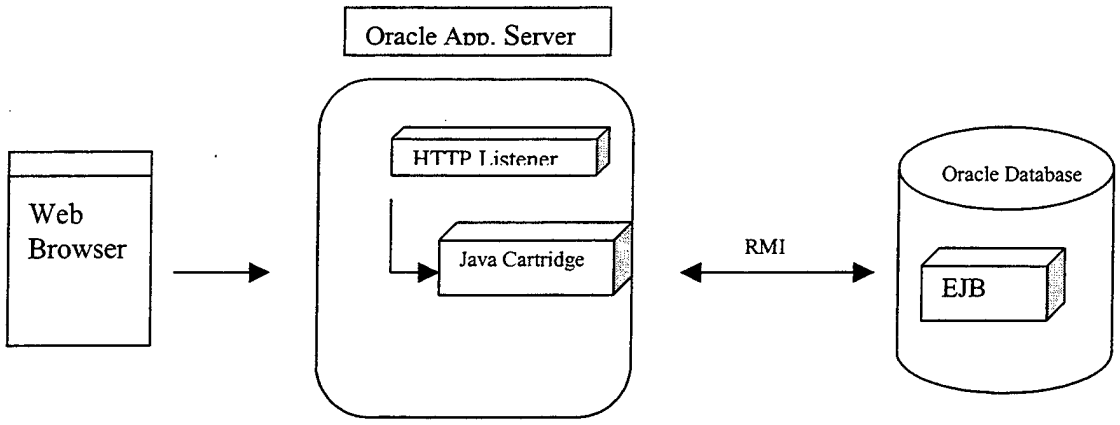



Figure 4.10 EJB Object in Oracle Database

Unlike client-side components, EJB (server-side) components are transactional, they encapsulate business logic, and they need to run on the server. An EJB executes in a container. A container provides an operating system process or thread in which to execute the components. Some types of containers are a web server, a transaction processing (TP) monitors and database systems. (William G., 2000, p. 219).

Enterprise Java Beans offer a higher level of abstraction than CORBA, because EJB does not require advanced systems programming skills, and they are simple for Java developers to develop like any other Java program.

***b) Common Object Request Broker Architecture (CORBA)***

CORBA is an object-oriented protocol that makes it possible for distributed programs written in different programming languages to communicate over a network, including the Internet. It is a standard for building, deploying, and managing distributed object applications that are interoperable across platforms. CORBA components written in different languages and running on different platforms can transparently communicate and interoperate. More precisely, the client and server code

can be written in any language (not just in Java), and are compiled into native machine code.

Oracle8i uses Java as its CORBA implementation language. Oracle8i integrates a Java-based CORBA that provides users with the ability to call into and out of the database using IIOP. Oracle8i comes with a complete set of tools for developing CORBA applications. Using these tools, developers can compile IDL specifications or load Java source files or classes into the database.

### *c) Java Stored Procedures*

Java stored procedures allow users to program the database by adding business rules to extend SQL. Java programs can be stored and executed in the Oracle database as Java stored procedures. Such procedures may use JDBC or SQLJ (Java with embedded SQL statement) to access data.

Java stored procedures are compiled once and stored in executable form, so procedure calls are quick and efficient. Executable code is automatically cached and shared among users. This reduces memory requirements and invocation overhead.

THIS PAGE INTENTIONALLY LEFT BLANK

# V. CONCLUSION

As I explained in detail in Chapter II, a rich variety of architecture and programming models can be used to implement a Web-Database application. For example, an Oracle or SQL Server can be chosen as the database, and ODBC or JDBC can be the connection between the database and the programming model. One of the programming techniques from Jsp, Java Servlets, Asp, etc. can be selected to provide the dynamic content of the web page. Tomcat, Resin, IIS or Web Logic can be selected to serve the static and dynamic web pages.

Selecting one model over another can be a difficult task. Each model has strengths and weaknesses for each particular application. Developers generally select the technique, with which they are most comfortable. Selecting a technique also depends on user perspectives, style, and priorities.

Coming to my thesis implementation, I spent a considerable amount of time to decide on my approach for implementing the Web-Database. It was obvious that there were many possible paths to follow. As a result of my research, I decided to use Oracle, Apache/Tomcat, Java (EJB, JSP, Java Script), and Windows NT 4.0. This section represents why I have chosen these methods.

## A. WHY ORACLE DATABASE SERVER?

First, Oracle is available on multiple platforms such as Windows, Linux, and all Unix platforms from vendors such as IBM, Sun, HP, etc. The multi-platform nature of Oracle makes it a true enterprise solution.

Oracle has multi-version consistency, which means that "readers do not block writers and writers do not block readers." In other words, the reader will see the data as it

was before the writer began changing it (until the writer commits). Oracle manages this scheme by creating a dynamic read-consistent image for a reader. Some other databases manage this scheme by locking the data, which results in a lot of delays.

In Oracle, the large tables and indexes can be partitioned at the database level. For example, a 10GB invoice table can be partitioned into monthly invoice table partitions. Such partitioned tables and partitioned indexes give performance and maintenance benefits, and are transparent to application.

One of the strongest features of the Oracle database is its ability to scale up for handling extremely large volumes of data and users. Oracle scales not only by running on more and more powerful platforms, but also by running in a distributed configuration. Oracle databases on separate platforms are combined to act as a single logical distributed database. (Stern J., 1999, p. 27)

Advanced replication and backup is another attractive feature of Oracle. Oracle comes with a whole core of functionality to keep the data accessible 24/7. Oracle backs up the data while the user community is still accessing it (hot backup). Moreover, by using the Advanced Replication feature, data can be replicated in another physical location. In case of any disaster, the replicated data can be used for recovery.

The usage of Internet and World Wide Web is extremely fast. Oracle provides strong e-business tools, which are integrated with the DBMS. Oracle is well positioned in this area with its Electronic Commerce Server, the Oracle Application Server and also other business planning tools.

Data warehousing has become one of the most powerful trends in information technology. There is a simple motivation behind this trend: data warehousing allows

businesses to use their data to aid in making statistical and strategic decisions. Oracle has added data warehousing related features to its DBMS. Oracle has also developed additional tools for building a complete data warehouse infrastructure, including business analysis and data movement tool.

Finally, Oracle is secure. The Oracle security model is a multi-layered one. It incorporates the protection of files and objects both inside and outside of the database, as well as a variety of administrative policies and technical strategies.

## B. WHY JAVA PROGRAMMING LANGUAGE?

Oracle has made a strategic commitment to Java by integrating it into a large portion of its product offerings. Oracle's complete Java platform provides an integrated set of products that enable the development, debugging, and deployment of database applications. The Oracle Java Platform consists of two related execution environments: a Java Virtual Machine (JVM) integrated with the Oracle database to run data intensive Java applications and a Java cartridge in the Oracle Application Server. Both share a common programming environment and programming interface.

In addition to the close relationship between Oracle and Java, Java has other advantages. Java is built on the principle of "Build once, run everywhere" and therefore can overcome cross-platform obstacles. In another words, because there is a standard, and because compiled Java byte code is portable across all platforms that support JVM, using Java does not lock you into using a specific hardware platform, operating system, or server software. For example, if a switch becomes necessary between components (hardware or software), all JSP pages and associated Java classes (EJB) can be migrated over as is. Moreover, Java web technologies provides:

- Robustness and Scalability (n-tier systems)

- Easy database access through JDBC
- Modularity - code reusability (Enterprise Java Beans)
- Separation of content from appearance (Servlets-JSP)

As an object-oriented language with strong typing, encapsulation, exception handling, and automatic memory management, the use of Java increases program productivity, and a more robust code.

Java Server Pages is a new technology to create web applications that connect to server-side Java components. As part of the Java family, it inherits all of the benefits of the Java language, including platform- and server-independence, a modular and reusable component architecture, and access to the rich family of Java API's.

JSP also offers a simpler mechanism for writing a small amount of program logic than a full-blown Servlet does. JSP scripting is a very powerful mechanism that provides the full power of Java in the simple form of scripting within an HTML page. In fact, developers can implement their entire application using only JSP, without ever writing a single explicit servlet, since JSP files are automatically translated into Java servlets when they are executed. JSP itself offers several advantages a system for dynamic content generation. Among these are improved performances over CGI, and a programming model that emphasizes component-centric application design.

Separation of presentation and implementation can be accomplished by using JSP technology. By taking advantage of Java Beans, it becomes possible to maintain a strict separation between data presentation (the display of information to the end user) and program implementation (the code used to generate that information in the first place).

The benefit of decoupling these two aspects is that changes to one can be made without requiring any changes to other.

JSP is based on a model in which JavaBeans and Enterprise JavaBeans (EJB) components contain the business and data logic for an application (component-centric platform). JavaBeans are Java's answer to Microsoft's ActiveX components. A JavaBean is the architecture for using and building components in Java.

EJB brings the component model of development to middleware. Enterprise middleware development is notoriously complicated; it involves not only business logic, but concurrency and scaling issues, as well as gluing together incompatible systems on incompatible platforms. Enterprise Java Beans solve middleware development complexity by factoring this entire infrastructure into containers. This allows the developer to focus on writing the business logic without having to worry about synchronization, scalability, transaction integrity, networking, distributed object frameworks, and other related matters.

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX A

In this appendix, I will present my demonstration and the code, which has been implemented using JDeveloper 3.1.1.2. My prototype is completely web enabled and currently published inside the school firewall so that the NPS community can access it via web browser and Internet connection (school account).

## A. THE WEB USER INTERFACE

Currently, users can access the NPS Bulletin Board from URL: http://131.120.179.222:7070/ugur/buildjsps/main.jsp

### 1. NPS Bulletin Board Main Page

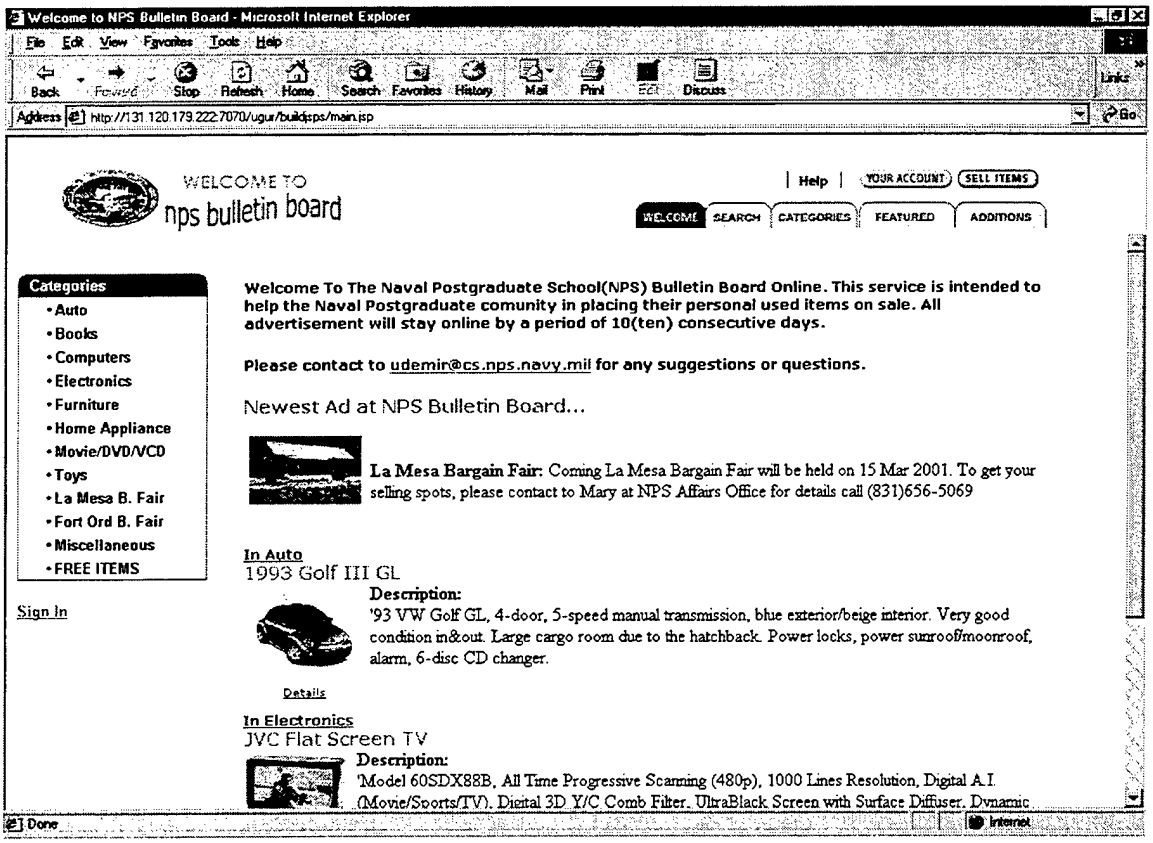



Figure A.1 Menu

## 2. Search by Category

In this menu, the user can search for an item by selecting a category

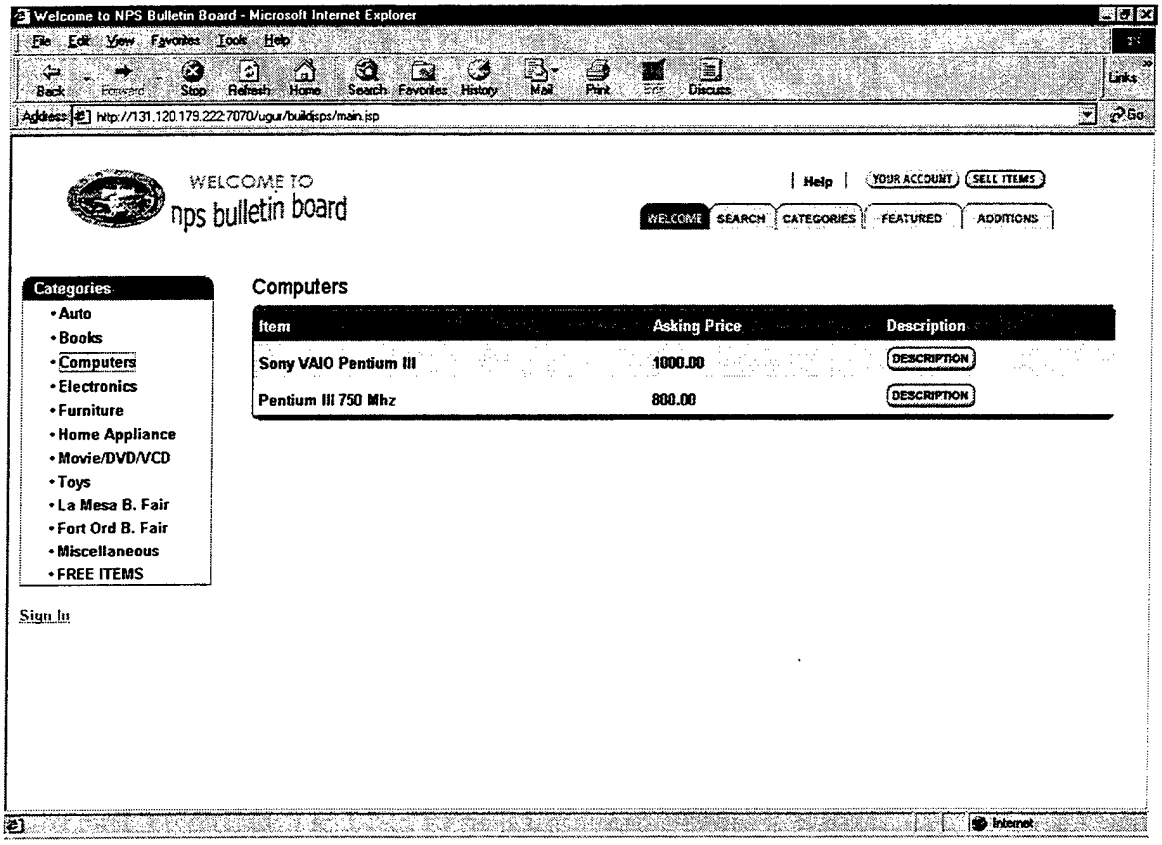



Figure A.2 Search by Category

Detailed information of the item can be viewed by clicking on "Description" button.

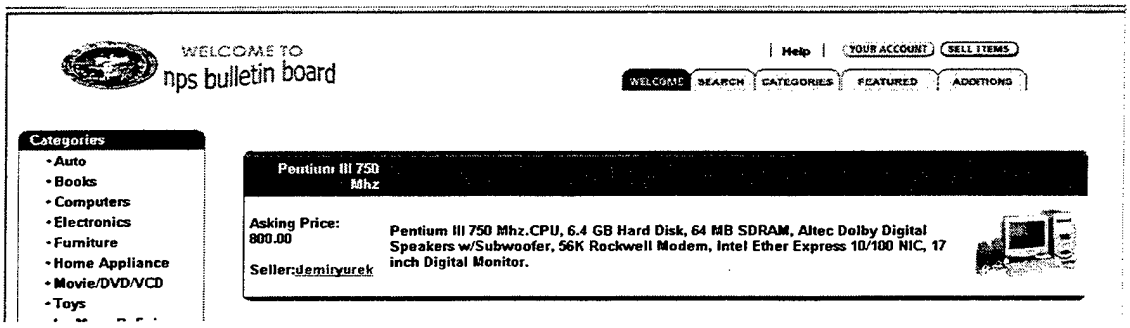



Figure A.3 Detail Information

### 3. Search Menu

The query given by the user is searched in the database and all matching records are returned.

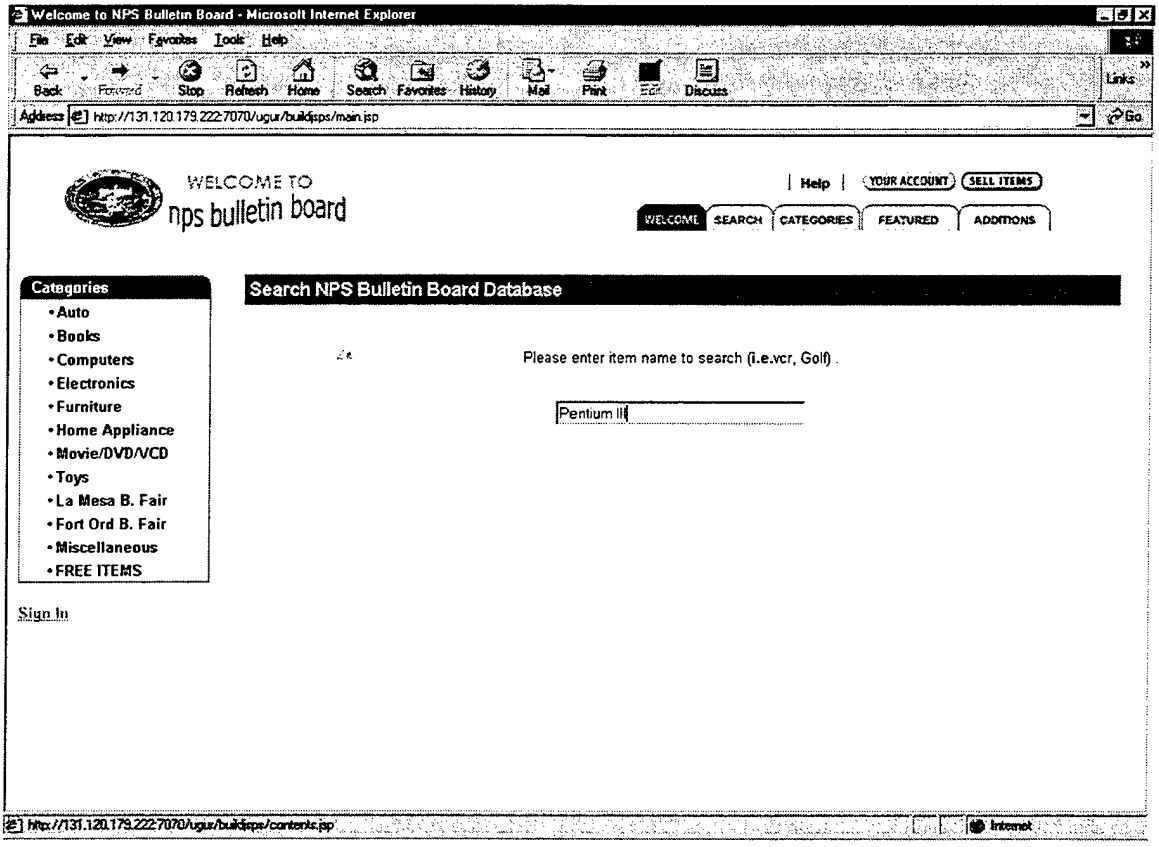



Figure A.4 Search Menu

## 4. Featured Items Menu

All items which have an image are viewed on this menu.

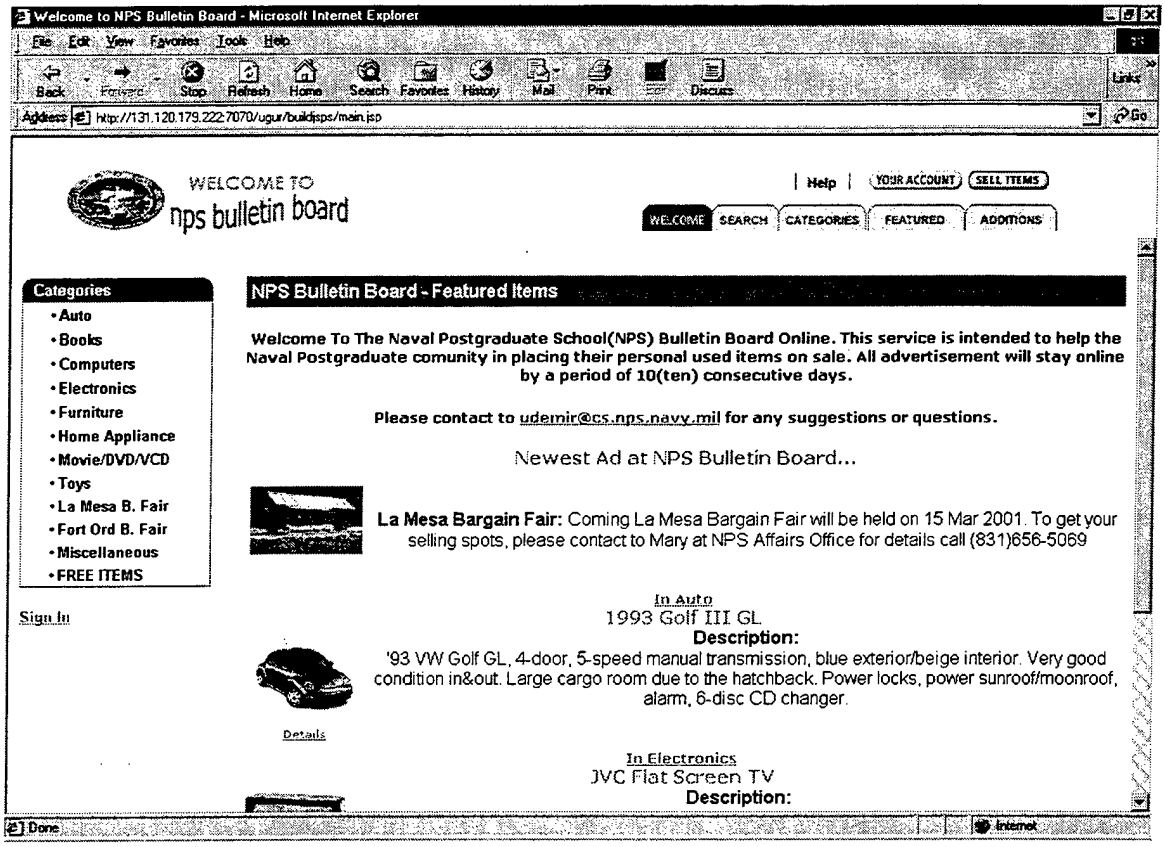



Figure A.4 Featured Items Menu

## 5. Sell Item Menu

User must sign in to sell an item

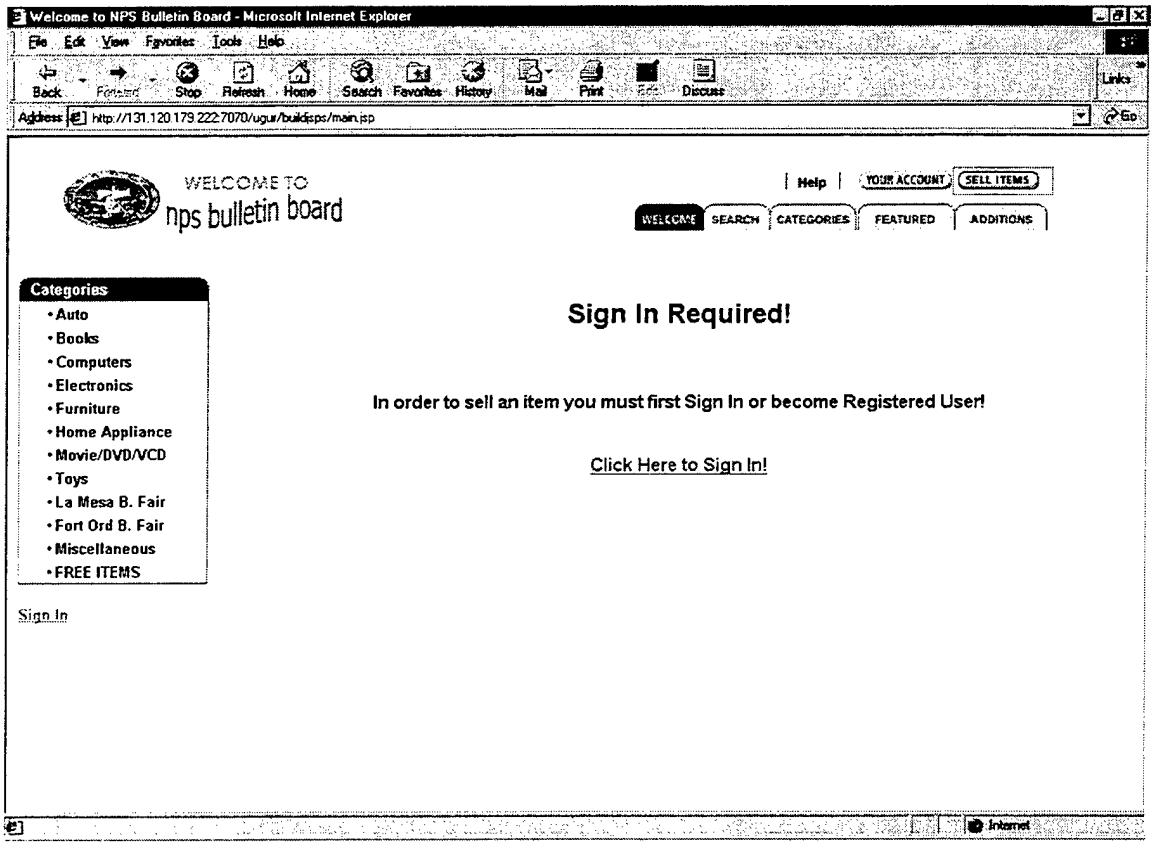



Figure A.5 Sell Item Menu

If the user is not a registered user, he can sign up to have an account for selling purposes. To sign up the user must enter his personal information.

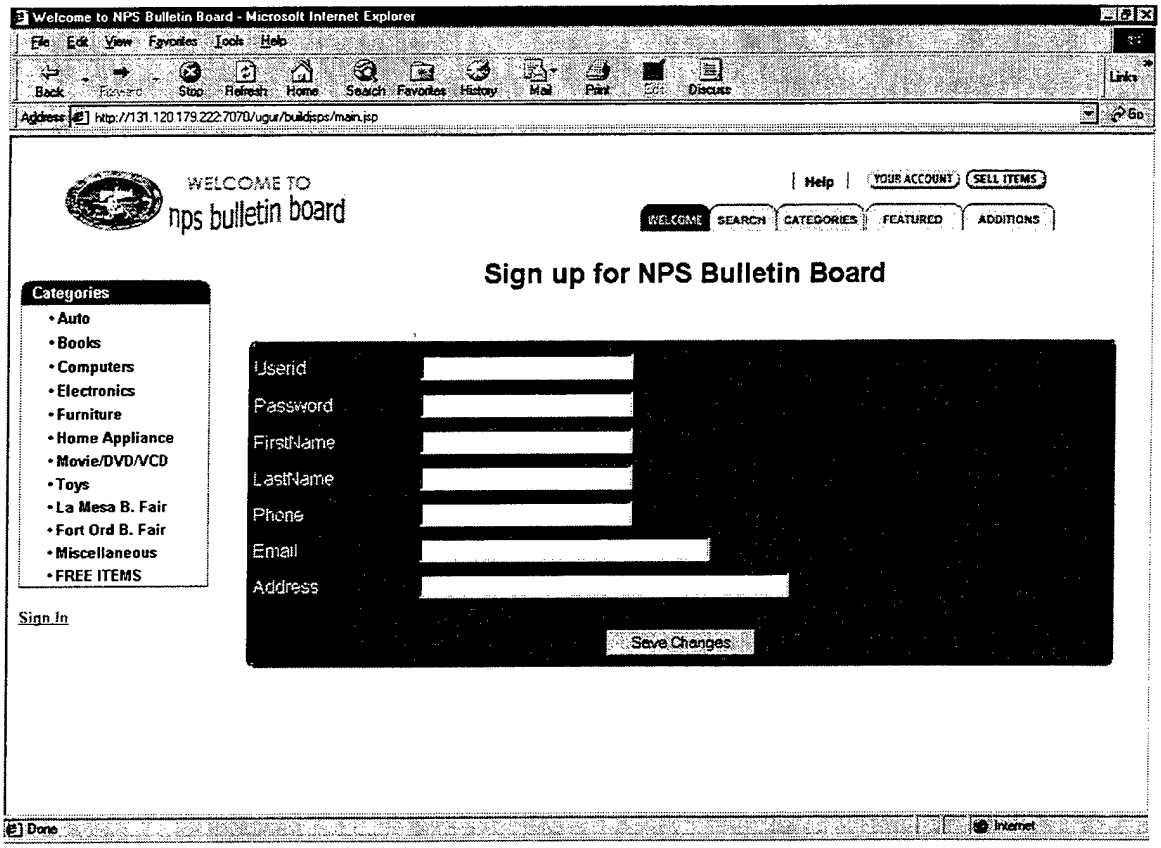



Figure A.6 User Information Entry Form

If the user is already registered, he has to supply a "User ID" and "Password" to sell his item.

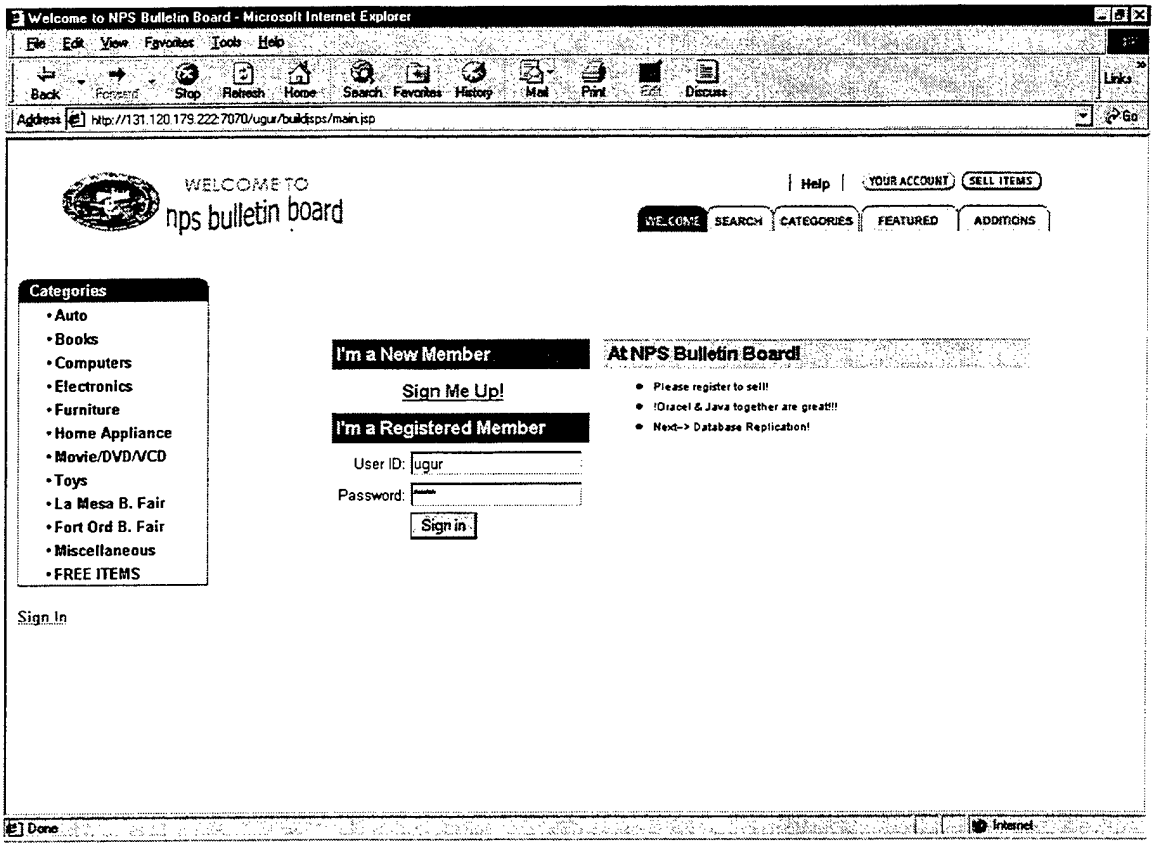



Figure A.7 User ID and Password Entry Menu

In the Greeting page, the user's First Name and Last Name are displayed. Also, under the Categories menu, two extra buttons are created for signing out and selling an item.

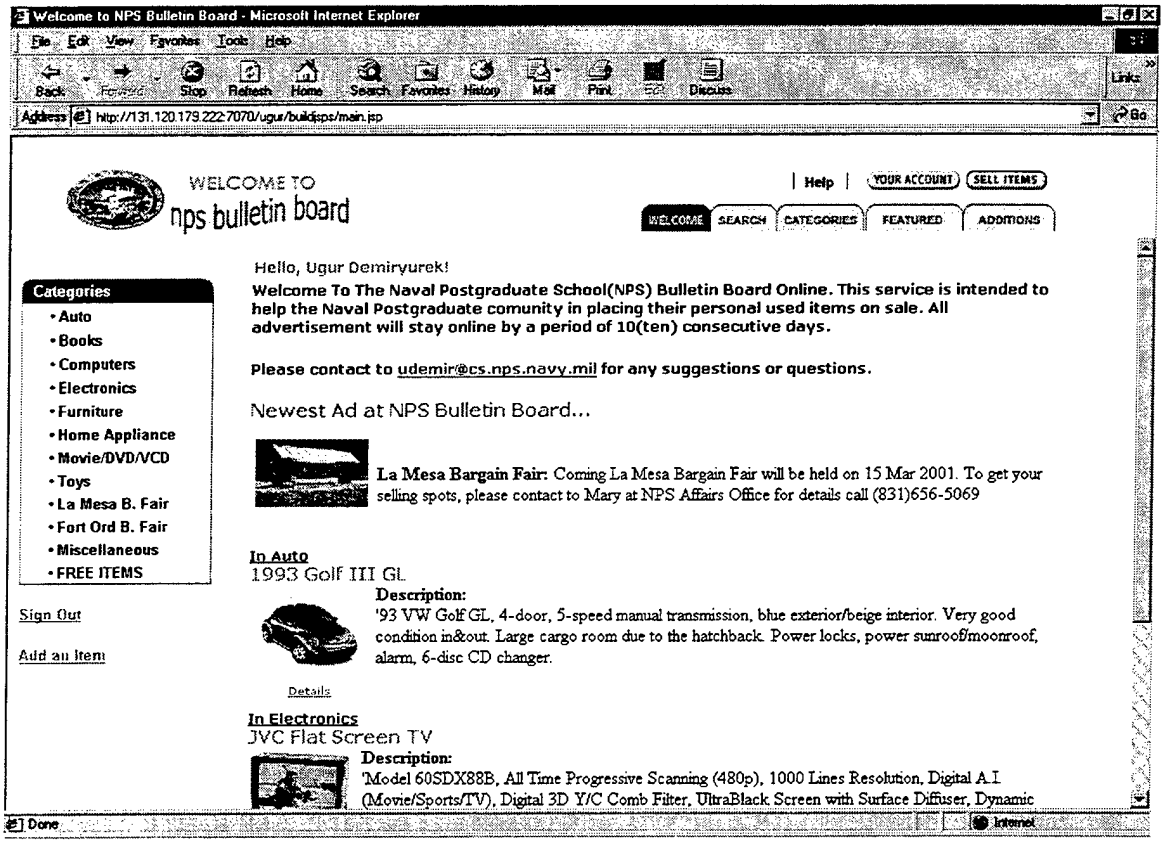



Figure A.8 Greeting Page

To advertise an item on the NPS Bulletin Board, the user enters his item's information.

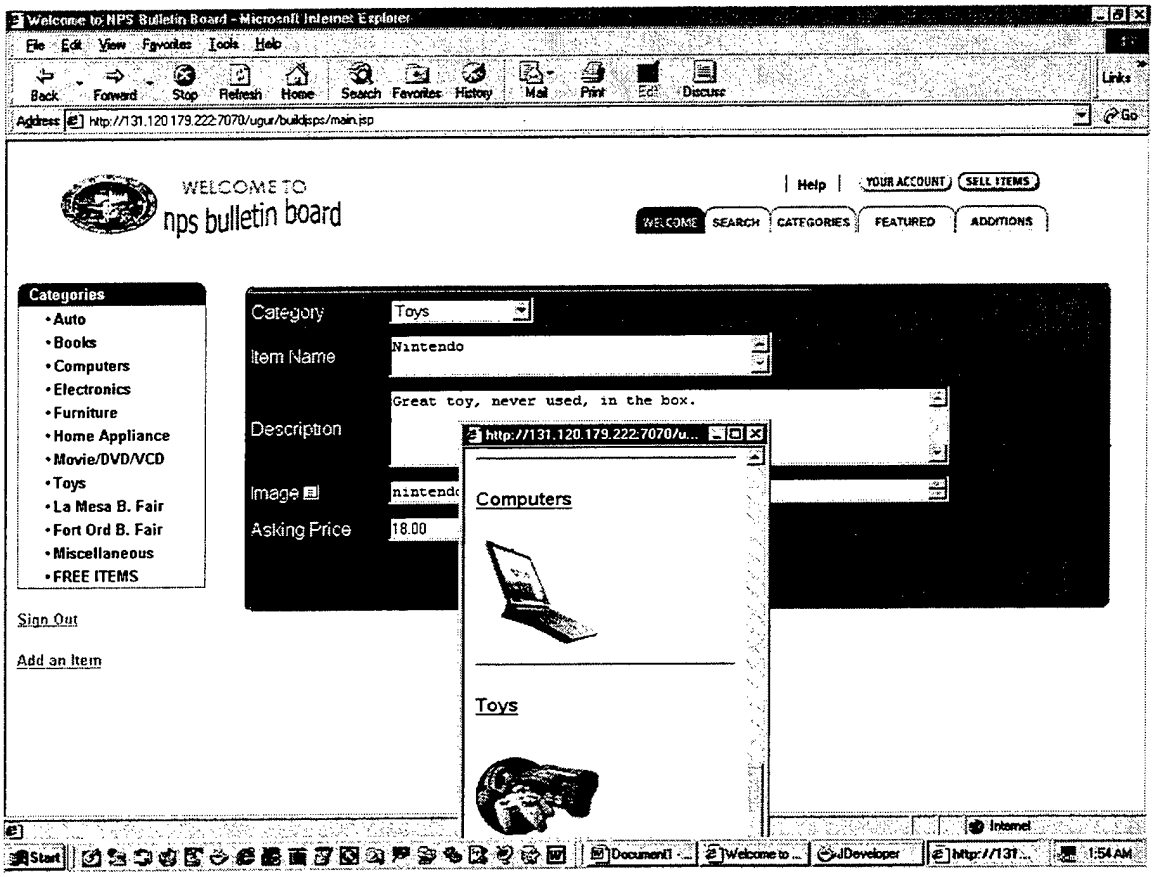



Figure A.9 Item Entry Form

## B. THE IMPLEMENTATION CODE

The implementation code of the NPS Bulletin Board is Java. JDeveloper is used to create the EJB and jsp pages. JDBC connection to Oracle database is also done via JDeveloper.

### 1. JDBC Connection

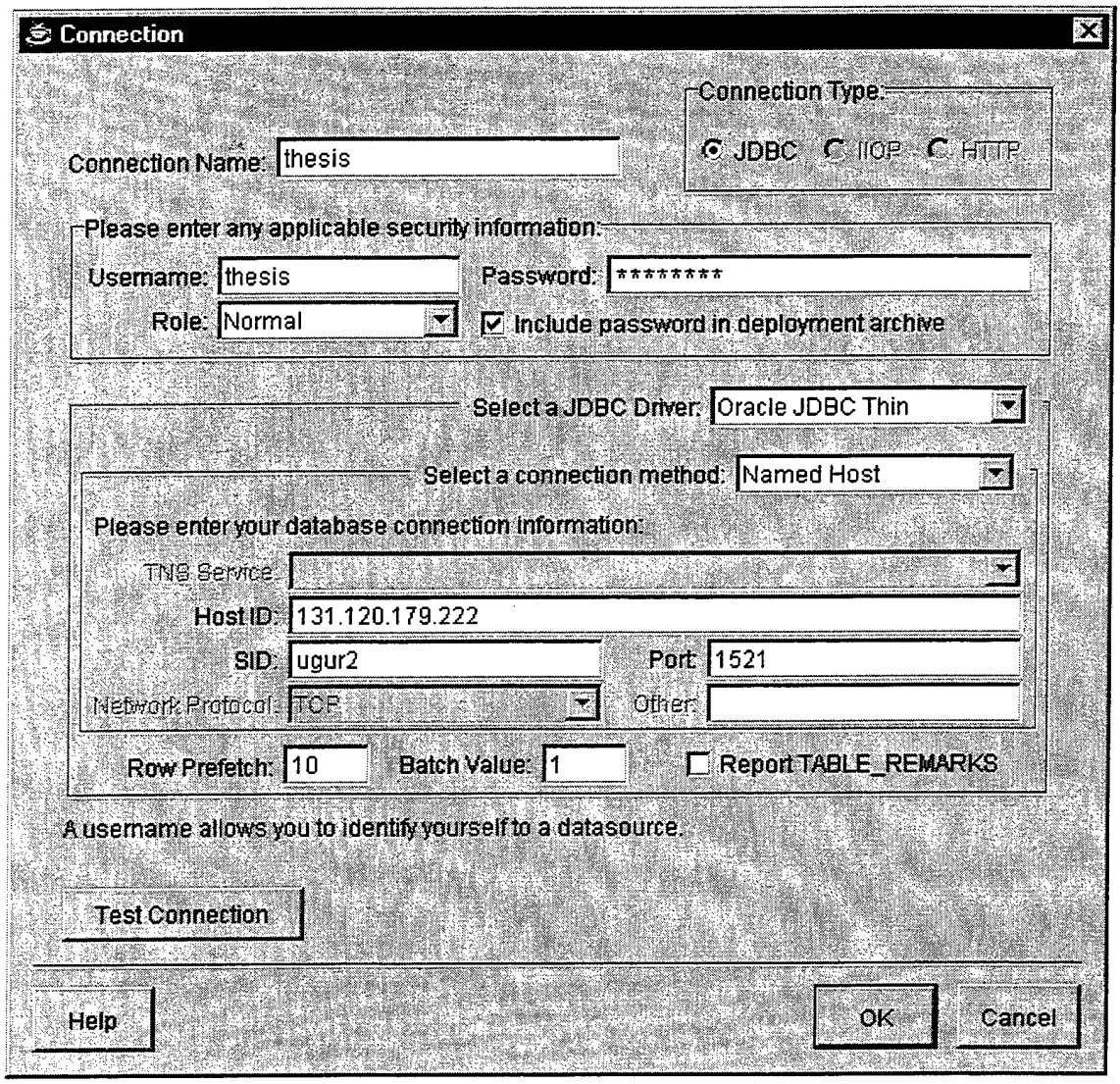



Figure B.1 JDBC Connection to Oracle Database

## 2. Enterprise Java Beans

- **featureBean.java**

```
import java.io.*;
import oracle.jbo.*;
import oracle.jdeveloper.html.*;

public class featureBean extends oracle.jdeveloper.html.DataWebBeanImpl {

 public void render() {
  try
  {

out.println( "<td> <!-- left panel  featured items -->");
out.println( "");
out.println( "<p>");
out.println( "<strong><font face=verdana,arial,helvetica size=2> Welcome To
The Naval Postgraduate School(NPS) Bulletin Board Online.");
out.println( "This service is intended to help the Naval Postgraduate comunity in
placing their personal used items on sale.");
out.println( "All advertisement will stay online by a period of 10(ten) consecutive
days.");
out.println( "<p>");
out.println( "<p>");
out.println( "Please contact ");
out.println( "<A
href=\"mailto:udemir@ cs.nps.navy.mil\">udemir@cs.nps.navy.mil</a>");
out.println(  "for any suggestions or questions.</font></strong>");
out.println( "</p>");
out.println( "<font face=verdana,arial,helvetica size=3
color=#FF0000><b>Newest Ad at NPS Bulletin Board...</b></font>");
out.println( "<br clear=left>");
out.println( "");
out.println( "");
out.println( "<table border=0 align=left cellpadding=0 cellspacing=0><tr><td>");
out.println( "<IMG alt=\"\" border=0
src=\"/ugur/webapp/images/old/la_mesa.gif\" ></a>");
out.println( "</td></tr><tr><td><center><font face=verdana,arial,helvetica
size=-2>");
out.println( "");
out.println( "");
out.println( "</font></center>");
out.println( "</td></tr></table> <p><b>La Mesa Bargain Fair:");
out.println( "</b> Coming La Mesa Bargain Fair will be held on 15 Mar 2001.");
```

```
        out.println( "  To get your selling spots, please contact to Mary at NPS Affairs
Office");
        out.println( "  for details call (831)656-5069");
        out.println( "  </p>");
        out.println( "<p>");
        out.println( "<br clear=left>");
        out.println( "");
        out.println( "<!-- item 3 --></p>");
        out.println( "<p>");
        out.println( "<font face=verdana,arial,helvetica size=-1><b>");
        out.println( "<A href=\"srch_results.jsp?QRY=cat_id=1\">");
        out.println( "In");
        out.println( "          Auto</b></font><br></A>");
        out.println( "<strong><font face=verdana,arial,helvetica color=#cc6600> 1993
Golf III GL</font></strong><br>");
        out.println( "");
        out.println( "");
        out.println( "");
        out.println( "<table border=0 align=left cellpadding=0 cellspacing=0><tr><td>");
        out.println( "<A href=\"BidsView_Browse.jsp?ITEMROWINDEX=2\">");
        out.println( "<IMG alt=\"BidsView_Browse.jsp?ITEMROWINDEX=2\" border=
src=\"/ugur/webapp/images/old/cars.gif\" ></a>");
        out.println( "</td></tr><tr><td><center><font face=verdana,arial,helvetica
size=-2>");
        out.println( "");
        out.println( "<A
href=\"BidsView_Browse.jsp?ITEMROWINDEX=50\">Details</a>");
        out.println( "                    ");
        out.println( "</font></center>");
        out.println( "</td></tr></table><b>Description:");
        out.println( "          </b> <br>'93 VW Golf GL, 4-door, 5-speed manual
transmision, blue exterior/beige interior. Very good condition in&out. Large cargo room
due to the hatchback. Power locks, power sunroof/moonroof, alarm, 6-disc CD
changer.");
        out.println( "</p>");
        out.println( "<p>        ");
        out.println( "      ");
        out.println( "  ");
        out.println( "");
        out.println( "<br clear=left>");
        out.println( "");
        out.println( "<!-- item 2 --></p>");
        out.println( "");
        out.println( "<p>");
        out.println( "<A href=\"srch_results.jsp?QRY=cat_id=4\">");
```

```
        out.println( "<font face=verdana,arial,helvetica size=-1><b>In
Electronics</b></font><br>");
        out.println( "</a>");
        out.println( "<strong><font face=verdana,arial,helvetica color=#cc6600> JVC
Flat Screen TV</font></strong><br>");
        out.println( "");
        out.println( "");
        out.println( "");
        out.println( "<table border=0 align=left cellpadding=0 cellspacing=0><tr><td>");
        out.println( "<A href=\"BidsView_Browse.jsp?ITEMROWINDEX=45\">");
        out.println( "<IMG alt=\"\" border=0  src=\"/ugur/webapp/images/old/video.gif\"
></a>";
        out.println( "</td></tr><tr><td><center><font face=verdana,arial,helvetica
size=-2");
        out.println( "");
        out.println( "<A
href=\"BidsView_Browse.jsp?ITEMROWINDEX=45\">Details</a>");
        out.println( "");
        out.println( "</font></center>");
        out.println( "</td></tr></table><b>Description:");
        out.println( "            </b> <br>'Model 60SDX88B, All Time Progressive
Scanning (480p), 1000 Lines Resolution, Digital A.I. (Movie/Sports/TV), Digital 3D Y/C
Comb Filter,  UltraBlack Screen with Surface Diffuser, Dynamic Focus.");
        out.println( "</p>");
        out.println( "<p>          ");
        out.println( "        ");
        out.println( "   ");
        out.println( "");
        out.println( "<br clear=left>");
        out.println( "<p>");
        out.println( "<br clear=left>");
        out.println( "");
        out.println( "");
        out.println( "<font face=verdana,arial,helvetica size=-1><b>");
        out.println( "<A href=\"srch_results.jsp?QRY=cat_id=3\">");
        out.println( "In");
        out.println( "            Computers</b></font><br></A>");
        out.println( "<strong><font face=verdana,arial,helvetica color=#cc6600> Pentium
III 750 Mhz.</font></strong><br>");
        out.println( "");
        out.println( "<table border=0 align=left cellpadding=0 cellspacing=0><tr><td>");
        out.println( "<A href=\"BidsView_Browse.jsp?ITEMROWINDEX=51\">");
        out.println( "<IMG alt=\"BidsView_Browse.jsp?ITEMROWINDEX=51\"
border=0  src=\"/ugur/webapp/images/old/desktop.gif\" ></a>");
        out.println( "</td></tr><tr><td><center><font face=verdana,arial,helvetica
size=-2>");
```

```
        out.println("<A
ref=\"BidsView_Browse.jsp?ITEMROWINDEX=51\">Details</a>");
        out.println( "                    ");
        out.println( "</font></center>");
        out.println( "</td></tr></table><b>Description:");
        out.println( "          </b> <br>'Pentium III 750 Mhz.CPU, 6.4 GB Hard Disk, 64
MB SDRAM, Altec Dolby Digital Speakers w/Subwoofer, 56K Rockwell Modem, Intel
Ether Express 10/100 NIC, 17 inch Digital Monitor.");
        out.println( "</p>");
        out.println( "<p>          ");
        out.println( "<br clear=left>");
        out.println( "");
        out.println( "</td>");
        out.println( "");
        out.println( "<!--   Left panel end -->");

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

- **greetings.java**

```
import java.io.*;
import oracle.jbo.*;
import oracle.jdeveloper.html.*;

public class greetings extends oracle.jdeveloper.html.WebBeanImpl {

 public void render() {

String loggedin = (String) session.getValue("ISLOGGEDIN");

if ( loggedin.equals("true") )

{
  out.println("<font face=verdana,arial,helvetica color=#CC6600 size=-1><b>");
  out.println(" Hello,");
  out.print(session.getValue("USERNAME") + " ");
  out.print(session.getValue("USERLASTNAME"));
  out.println("!</font></b><br>");
 }
 else {
   out.println("<br>");
   }
 }
public void renderlogin(){

String loggedin = (String) session.getValue("ISLOGGEDIN");
 if ( loggedin.equals("false") )

{
  out.println("<p><font face=\"Arial\"  size=2 color=\"blue\"  >");
  out.println("<a href=\"login.jsp\" target=\"contentsFrame\"><b>Sign
In</b></a>");

  out.println("</font>");
}
else {
  out.println("<p><font face=\"Arial\"  size=2 color=\"blue\"  >");
  out.println("<a href=\"signout.jsp\" target=\"_top\"><b>Sign Out</b></a>");
  out.println("</font>");
  out.println("<p><font face=\"Arial\" size=2 color=\"blue\"> ");
  out.println("<a href=\"ItemsView_Insert.jsp\"
target=\"contentsFrame\"><b>Add an Item</b></a></font>");

  }
```

- **loginBean.java**

```
import java.io.*;
import java.io.PrintWriter;
import oracle.jbo.*;
import oracle.jdeveloper.html.*;

public class loginBean extends oracle.jdeveloper.html.DataWebBeanImpl {

String userid = "";
String password = "";

//Set methods

 public void setLoginParms(String p_userid, String p_password){
   userid = p_userid;
   password = p_password;
  }

 public  void execute() {

  Row resultRow;

 // get the rowset instance
 RowSet         qView = getRowSet();
  qView.getViewObject().setWhereClause(" userid = '" + userid + "' AND
password = '" + password + "'");

  try
 {
  qView.executeQuery();
  qView.first();

  // User Validated
  // Set session variables

  resultRow = qView.getCurrentRow();

  session.putValue("USERNAME", resultRow.getAttribute( "FirstName"
).toString());
  session.putValue("USERLASTNAME", resultRow.getAttribute( "LastName"
).toString());
  session.putValue("CUSTID", resultRow.getAttribute("Id").toString() );
  session.putValue("EMAIL", resultRow.getAttribute("Email").toString() );
  session.putValue("ISLOGGEDIN", "true");
```

```
        // print header with meta refresh tag to main.jsp
            out.println("<META HTTP-EQUIV=\"refresh\" CONTENT=\"1;
URL=main.jsp\"> ");
            out.println("</head><body>");
            out.println("<br><center><h2>Logging In.....</h2></center>");


          }
                                        catch(Exception ex)
           {
        // print header with meta refresh tag to login.jsp
            out.println("<META HTTP-EQUIV=\"refresh\" CONTENT=\"1;
URL=login.jsp\"> ");
            out.println("</head><body>");
            out.println("<br><center><h2>Invalid Login, try again.</h2></center>");

            if(ex.getMessage() != null)
              out.println(ex.getMessage());

            return;

          }// catch

         } // execute

        }
```

- **qpBean.java**

```
import java.io.*;
import java.io.PrintWriter;
import java.util.Vector;
import oracle.jbo.*;
import oracle.jdeveloper.html.*;

public class qpBean extends oracle.jdeveloper.html.DataWebBeanImpl {

String yellow_dot = " <font color=\"#FFFF00\">&#149;</font>";
String red_dot = " <font color=\"#FF0000\">&#149;</font>";
int font_size = 2;
String font_color ="#000000";
String font_face = "Arial, Helvetica, sans-serif";
String qp_display[];
String qp_link[];
int linksnum = 0;
int i;

  AttributeDef[] attrs;
   Row[] rows;

//Set methods

 public void set_font_size(int size){
   font_size = size;
  }

 public void set_font_color(String color){
   font_color = color;
  }

 public void set_font_face(String face){
   font_face = face;
  }

 public  void render() {

  AttributeDef[] attrs;
  Row[] rows;

  try   {

// Retrieve all records by default
  qView.setRangeSize(-1);
```

```
            qView.first();

            rows = qView.getAllRowsInRange();
            attrs = qView.getViewObject().getAttributeDefs();

            linksnum = rows.length;

          // print table wrapper
            out.print("<!-- JSP Generated QUICK PICK --> \n"+
                  "<table width=\"170\" cellpadding=0 cellspacing=0 border=0>"+
                  "<tr valign=\"top\">"+
                  "<td width=\"10\"><img src=\"/ugur/webapp/images/hdr_left.gif\"
width=10 height=20 border=0 align=\"top\"></td>"+
                  "<td width=\"150\" valign=\"middle\" bgcolor=\"#006699\" nowrap><font
face=\"Arial,Helvetica,sans-serif\" size=\"2\"
color=\"#ffffff\"><b>Categories</b></font></td>"+
                  "<td width=\"10\" align=\"right\"><img
src=\"/ugur/webapp/images/hdr_right2.gif\" width=10 height=20 border=0
align=\"top\"></td>"+
                  "</tr></table>"+
                  "<table width=\"170\" cellpadding=1 cellspacing=0 border=0
bgcolor=\"#006699\">"+
                  "<tr><td><table width=\"100%\" cellpadding=0 cellspacing=0 border=0
bgcolor=\"#FFFFFF\">"+
                  "<tr><td width=\"10\"><font face=\"Arial,Helvetica,sans-serif\"
size=\"2\"> </font></td><td >");


          out.print("<!-- JSP Generated QUICK PICKS MENU --> \n"+
                "<TABLE BORDER=\"0\" CELLSPACING=\"0\" WIDTH=\"143\"
HEIGHT=\"83\" >\n" );

          if ( linksnum < 1 ) {
            out.println("<tr><td>No links added! Use addLink!</td></tr>\n");
          }
          else {
           for ( i=0; i<linksnum; i++){
            out.println("<TR><TD WIDTH=\"15%\" ALIGN=\"RIGHT\"
VALIGN=\"MIDDLE\">\n"+ red_dot + "\n" +
              "</TD>\n"); // print dot

            out.println("<TD WIDTH=\"85%\" ALIGN=\"LEFT\"
VALIGN=\"MIDDLE\">\n"+
              "<B><A HREF=\"srch_results.jsp?QRY=cat_id=" + rows[i].getAttribute( 0
).toString()  + "&cat_name=" + rows[i].getAttribute( 1 ).toString() + "\"
target=contentsFrame><SPAN STYLE=\"Text-Decoration : None\">\n"+
```

```
            "<FONT SIZE=\"" + font_size +"\" COLOR=\"" + font_color +"\" FACE=\""
+ font_face + "\">" + rows[i].getAttribute( 1 ).toString() +
"</FONT></SPAN></A></B>\n"+
            "</TD></TR>\n");

        } // for loop
       }  // else

        out.println(  "</TABLE>\n"+
          "</td></tr></table></td></tr></table><!-- END JDEV Generated QUICK
PICKS MENU -->");


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

- **srchresBean.java**

```
import java.io.*;
import oracle.jbo.*;
import oracle.jdeveloper.html.*;
import oracle.jbo.html.databeans.*;

public class srchresBean extends oracle.jdeveloper.html.DataWebBeanImpl {

 public void render() {
  try
  {

   oracle.jbo.html.databeans.RowsetNavigator  rsn;

   oracle.jbo.html.databeans.FindForm  find;

   oracle.jbo.html.databeans.RowSetBrowser srch;

   String catName = request.getParameter("cat_name");

   find= (oracle.jbo.html.databeans.FindForm) new
oracle.jbo.html.databeans.FindForm();

   rsn = (oracle.jbo.html.databeans.RowsetNavigator) new
oracle.jbo.html.databeans.RowsetNavigator();

   rsn.initialize(application,session,
request,response,out,"package1_Package1Module.srchItemsView");

   String QueryArg = request.getParameter("QRY");
   String ItemIndex = request.getParameter("ITEMROWINDEX");

  if ( QueryArg != null ) {
  // Using Category Bean
  rsn.getRowSet().getViewObject().setWhereClause(QueryArg);
   rsn.getRowSet().getViewObject().executeQuery();
   rsn.getRowSet().first();
  }

 if ( QueryArg == null && ItemIndex == null ) {
  // Using FindForm Search window
 find.initialize(application,session,
request,response,out,"package1_Package1Module.srchItemsView");
 find.execute();
  }
```

```
            // print category
            if ( catName != null && catName != "" )
            {
            out.println("<p> <font face=\"Arial\"><big>" + catName +
"</big></font><br>");
            session.putValue("CATNAME", catName);
            }
            else
             out.println("<br>");

           // rowset browser bean
            srch = (oracle.jbo.html.databeans.RowSetBrowser) new
oracle.jbo.html.databeans.RowSetBrowser();

            srch.initialize(application,session,
request,response,out,"package1_Package1Module.srchItemsView");
            srch.setVisibleRows(100);
            srch.setShowCurrentRow(false);
            srch.setDisplayAttributes("ItemName,StartPrice");
            srch.setAttributeTitle("ItemName" , "Item" );
            srch.setAttributeTitle("StartPrice" , "Asking Price" );
            srch.addImageUrlColumn("Description", "/ugur/webapp/images/desc.jpg",
"BidsView_Browse.jsp?BIDVIEW=srchBidsView", "contentsFrame" );
            srch.render();

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

### 3. Java Server Pages

- **main.jsp**

```
<%@ page contentType="text/html;charset=WINDOWS-1252"%>
<HTML>
<HEAD>
<title>Welcome to NPS Bulletin Board</title>
</HEAD>

<%
// Check to see if logged in already.

String loggedin = (String) session.getValue("ISLOGGEDIN");

if ( loggedin == null )
{
session.putValue("ISLOGGEDIN", "false");
}
%>

<!-- Define Frameset -->

<FRAMESET ROWS="90,*" FRAMESPACING="0" FRAMEBORDER="0"
border="false" >
<FRAME SRC="title.html" NAME="titleFrame"  FRAMEBORDER="0"
MARGINHEIGHT=1
MARGINWIDTH=1 FRAMEBORDER="0" FRAMESPACING="0"
SCROLLING="No">
<FRAMESET cols="200,*"
FRAMESPACING="0" FRAMEBORDER="0" SCROLLING="AUTO">
<FRAME SRC="nav.jsp" name="navFrame">
<FRAME src="contents.jsp" name="contentsFrame">
</FRAMESET>
</FRAMESET>


</HTML>
```

- **Login.jsp**

```
<%@ page contentType="text/html;charset=WINDOWS-1252"%>
<%@ page language = "java" errorPage="errorpage.jsp" import = "java.util.*,
oracle.jbo.*, javax.naming.*, oracle.jdeveloper.html.*, oracle.jbo.common.appmgr.*,
oracle.jbo.html.databeans.*" %>


<%
// make sure the application is registered
oracle.jbo.html.jsp.JSPApplicationRegistry.registerApplicationFromPropertyFile(
session, "package1_Package1Module");
%>

<html>
<head>
<LINK REL=STYLESHEET TYPE="text/css"
HREF="<%=session.getValue("CSSURL")%>">

  <title>Welcome to NPS Bulletin Board</title>
</head>
<body bgcolor="#FFFFFF" TEXT="black" BGCOLOR="#FFFFFF"
LINK="#336699" ALINK="#6699CC" >
<p><br>
 <br> 
<br> 
<center><table BORDER=0 CELLSPACING=0 CELLPADDING=2
WIDTH="80%" >
<tr>
<td VALIGN=TOP WIDTH="1%">
<table BORDER=0 CELLSPACING=0 CELLPADDING=4 >
<tr>
<td BGCOLOR="#336699">
<b><font face="arial"><font color="#FFFFFF">I'm a New
Member</font></font></b>
</td>
</tr>
<tr>
<td ALIGN=CENTER>
<table BORDER=0 CELLSPACING=0 CELLPADDING=6 >
<tr>
<td><b>
<font face="Arial,Helvetica"><a href="CustomersView_Insert.jsp">Sign Me
Up!</a></font></b>
```

```
        <td>
        </tr>
        </table>
        </td>
        </tr><tr> <td NOWRAP BGCOLOR="#336699"><b><font face="arial"><font
color="#FFFFFF">I'm a Registered Member</font></font></b></td></tr>
        <form action="login_submit.jsp" target="_top"  method="post">

        <tr>  <td ALIGN=RIGHT> <table BORDER=0 CELLSPACING=0
CELLPADDING=2 >
        <tr>  <td ALIGN=RIGHT NOWRAP><font face="arial"><font size=-1>User
ID:</font></font></td>
        <td><input name="userid" size=20 maxlength=32 value=""></td></tr>

        <tr><td ALIGN=RIGHT NOWRAP><font face="arial"><font size=-
1>Password:</font></font></td>
        <td><input name="password" type="password" size=20 maxlength=32
value=""></td></tr>

        <tr><td> </td><td><input type="submit" value="Sign
in"></td></form></tr> </table></td> </tr> </table></td><td> </td><td
VALIGN=TOP WIDTH="100%"><table BORDER=0 CELLSPACING=0
CELLPADDING=4 WIDTH="100%" >

        <tr><td BGCOLOR="#CCCCCC"><b><font face="Arial">At NPS Bulletin
Board!</font></b></td></tr>
        <tr><td VALIGN=TOP><ul><li><font face="helvetica,arial"><font size=-
2>Please register to sell!</font></font></li>
        <li><font face="helvetica,arial"><font size=-2>!Oracel & Java together are
great!!!</font></li>
        <li> <font face="helvetica,arial"><font size=-2>Next--> Database
Replication!</font></font></li></ul> </td> </tr> </table> </td> </tr>
</table></center><br> <br> <center>
        <table BORDER=0 CELLSPACING=0 CELLPADDING=4 WIDTH="80%" >
        <tr>
        <td align="right"></td>
        </tr></table>
        </center>
        </body> </html>
```

- **Login_submit.jsp**

```
<%@ page contentType="text/html;charset=WINDOWS-1252"%>
<%@ page language = "java" errorPage="errorpage.jsp" import = "java.util.*,
oracle.jbo.*, javax.naming.*, oracle.jdeveloper.html.*, oracle.jbo.common.appmgr.*,
oracle.jbo.html.databeans.*" %>

<html>
<head>
<LINK REL=STYLESHEET TYPE="text/css"
HREF="<%=session.getValue("CSSURL")%>">


<jsp:useBean class="auctionbeans.loginBean" id="login" scope="request" >
<%
login.initialize(application,session,
request,response,out,"package1_Package1Module.CustomersView");

login.setLoginParms(request.getParameter("userid"),request.getParameter("password") );
login.execute();
%>
</jsp:useBean>

<br>


</body>
</html>
```

- **ItemView_Insert.jsp**

```
<%@ page contentType="text/html;charset=WINDOWS-1252"%>
<%@ page language = "java" errorPage="errorpage.jsp" import = "java.util.*,
oracle.jbo.*, javax.naming.*, oracle.jdeveloper.html.*, oracle.jbo.common.appmgr.*,
oracle.jbo.html.databeans.*" %>

<%
// make sure the application is registered
oracle.jbo.html.jsp.JSPApplicationRegistry.registerApplicationFromPropertyFile(
session , "package1_Package1Module");
%>

<html>
<head>
<SCRIPT LANGUAGE="JavaScript1.1">

</SCRIPT>

<LINK REL=STYLESHEET TYPE="text/css"
HREF="<%=session.getValue("CSSURL")%>">
</head>
<body>

<br>

<jsp:useBean id="RowEditor"
class="oracle.jbo.html.databeans.EditCurrentRecord" scope="request">
<%
String loggedin = (String) session.getValue("ISLOGGEDIN");

 if ( loggedin != null && loggedin.equals("true") )

{
%>

<%
RowEditor.initialize(application, session , request, response, out,
"package1_Package1Module.srchItemsView");

RowEditor.setTargetUrl("ItemsView_SubmitInsertForm.jsp?SellerId=" +
session.getValue("CUSTID"));


RowEditor.setDisplayAttributes("CatId,ItemName,Description,ImageUrl,StartPrice");
```

```
        RowEditor.createNewRow();

        RowEditor.useComboBox("CatId", "select id, name from categories", "NAME",
"ID" );
        RowEditor.getFieldRenderer("CatId").setPromptText("Category");

        RowEditor.getFieldRenderer("ItemName").setDisplayWidth(40);
        RowEditor.getFieldRenderer("ItemName").setDisplayHeight(2);
        RowEditor.getFieldRenderer("ItemName").setPromptText("Item Name");

        RowEditor.getFieldRenderer("Description").setDisplayWidth(60);
        RowEditor.getFieldRenderer("Description").setDisplayHeight(4);

        RowEditor.getFieldRenderer("ImageUrl").setDisplayWidth(60);
        RowEditor.getFieldRenderer("ImageUrl").setDisplayHeight(1);
        RowEditor.getFieldRenderer("ImageUrl").setPromptText("Image <A
HREF=\"javascript:imagepick()\"><img src=\"/ugur/webapp/images/itempick.gif\"
border=0></a>");

        RowEditor.getFieldRenderer("StartPrice").setPromptText("Asking Price");

        RowEditor.render();

       %>
       <%
       }
       else {
          out.println("<br><center><h2>Sign In Required!</h2></center><br>");
          out.println("<center><br><b>In order to sell an item you must first Sign In or
become Registered User!<br>");
          out.println("<br><br><a href=\"login.jsp\">Click Here to Sign
In!</a></b></center>");
          }
       %>

       </jsp:useBean>

       </body>
       </html>
```

- **CustomerView_Insert.jsp**

```
<%@ page contentType="text/html;charset=WINDOWS-1252"%>
<%@ page language = "java" errorPage="errorpage.jsp" import = "java.util.*,
oracle.jbo.*, javax.naming.*, oracle.jdeveloper.html.*, oracle.jbo.common.appmgr.*,
oracle.jbo.html.databeans.*" %>

<%
// make sure the application is registered
oracle.jbo.html.jsp.JSPApplicationRegistry.registerApplicationFromPropertyFile(
session , "package1_Package1Module");
%>

<html>
<head>
<LINK REL=STYLESHEET TYPE="text/css"
HREF="/ugur/webapp/css/auctions.css">
</head>
<body>

<center><h2>Sign up for NPS Bulletin Board</h2></center>


<br>
<jsp:useBean id="RowEditor"
class="oracle.jbo.html.databeans.EditCurrentRecord" scope="request">
<%
RowEditor.initialize(application, session , request, response, out,
"package1_Package1Module.CustomersView");
RowEditor.setTargetUrl("CustomersView_SubmitInsertForm.jsp?Id=999");


RowEditor.setDisplayAttributes("Userid,Password,FirstName,LastName,Phone,E
mail,Address");

RowEditor.getFieldRenderer("Userid").setDisplayWidth(20);
RowEditor.getFieldRenderer("Password").setDisplayWidth(20);


RowEditor.getFieldRenderer("FirstName").setPromptText("Asking Price");


RowEditor.getFieldRenderer("LastName").setDisplayWidth(20);
```

```
RowEditor.getFieldRenderer("Email").setDisplayWidth(30);
RowEditor.getFieldRenderer("Phone").setDisplayWidth(30);
RowEditor.getFieldRenderer("Address").setDisplayWidth(30);
RowEditor.getFieldRenderer("Address").setDisplayHeight(1);
RowEditor.createNewRow();
RowEditor.render();
%>
</jsp:useBean>

</body>
</html>
```

- **CustomerView_SubmitInsertForm.jsp**

```
<%@ page contentType="text/html;charset=WINDOWS-1252"%>
<%@ page language = "java" import = "java.util.*, oracle.jbo.*, javax.naming.*,
oracle.jdeveloper.html.*, oracle.jbo.common.appmgr.*, oracle.jbo.html.databeans.*" %>

<%
// make sure the application is registered
oracle.jbo.html.jsp.JSPApplicationRegistry.registerApplicationFromPropertyFile(
session, "package1_Package1Module");
%>

<jsp:useBean id="RowEditor"
class="oracle.jbo.html.databeans.EditCurrentRecord" scope="request">
<%
 RowEditor.initialize(application, session , request, response, out,
"package1_Package1Module.CustomersView");
 RowEditor.execute();
%>
</jsp:useBean>

<html>
<head>
<LINK REL=STYLESHEET TYPE="text/css"
HREF="/ugur/webapp/css/auctions.css">

<%

   Row row = RowEditor.getRowSet().getCurrentRow();

%>

<META HTTP-EQUIV="refresh" CONTENT="1; URL=login.jsp">
</head>
<body>

<center><h2>Customer Successfully Added!</h2>
<P>Proceeding to login page...</center>

<br>


</body>
</html>
```

THIS PAGE INTENTIONALLY LEFT BLANK

## LIST OF REFERENCES


1. Abbey, M., and others, *Oracle 8i Beginners Guide*, Osborne McGraw-Hill, 1999.
2. Bradley, D., *Oracle 8i Web Development*, Osborne McGraw-Hill, 2000.
3. Burlenson, D., *Oracle8 Tuning*, Coriolis Group, 1999.
4. Elmasri, R. and Navathe S., *Fundamentals of Database Systems, 3 rd. Edition*, Addison-Wesley, 2000.
5. Hernandez, M., *Database Design for Mere Mortals*, Addison-Wesley, 1998.
6. Horstman, C. and Cornell G., *Core Java 2*, Sun Microsystems Press, 1999.
7. JavaSoft Web site, "Remote Method Invocation Specification", [http://www.javasoft.com/products/jdk/1.1/docs/guide/rmi/spec/rmiTOC.doc.html]. May 1999.
8. JavaSoft Web site, "Press Archive", [http://javasoft.com/pr/archive/pr_2001.html?frontpage-spotlight]. Oct. 1999.
9. JavaWorld Web site, "Write a session EJB", [http://www.javaworld.com/javaworld/jw-07-1998/jw-07-step.html]. Jan. 2000.
10. Kolb M. and Duanne K., *Web Development with Java Server Pages*, Manning Press, 1999.
11. Kroenke David M., *Database Processing*, Prentice Hall, 1998.
12. Loney K., *Oracle DBA Handbook*, Oracle Press., 1999.
13. Lozano F., "Accessing Databases Using Java and JDBC" [http://www.edm2.com/0607/msql3.html]. April 2000.
14. Microsoft Web site, "ODBC Architecture.", [http://msdn.microsoft.com/library/psdk/dasdk/odch6gh1.htm]. November 200.
15. Morrison J. and Morrison M., "*A Guide to Oracle 8*", Course Technology, 2000.
16. Odewahn, A., *Oracle Web Applications*, O'reilly Press, September 1999.
17. Olivia R., "Advanced Topics in Database Sys.", [http://misdb.bpa.arizona.edu/~96g.]. Dec. 1998.
18. Oracle Corporation, "Deploying Enterprise JavaBeans in the Oracle 8i Server", Technical White Paper, 1999.

19. Oracle Web site, "General Documentation, Release 3 (8.1.7)", [http://otn.oracle.com/docs/products/oracle8i/doc_index.htm]. March 1999.
20. Oracle Web site, "Oracle 8i Enterprise Edition", [http://otn.oracle.com/software/products/oem/software_index.htm]. Dec. 2000.
21. Ramakrishan, R. and Gehrke, J., *Database Management Systems*,
22. Rick, G., Stackowiak, R. and Stern J., *Oracle Essentials, Oracle 8 & Oracle 8i*, O'Reilly Press, 2000.
23. Sun Systems Web site, "Java 2 Platforms Enterprise Documentation", [http://developer.java.sun.com/developer/infodocs/]. March 1999.
24. Sun Web site, "Core Servlets & JavaServer Pages". [http://www.sun.com/books/catalog/hall/]. Jan. 2000.
25. Tower, J. and Billings, M., *Rapid Application Development with Oracle Designer*, Addison-Wesley, 1997.
26. WebBase Web Site, "How WebBase Works?", [http://www. webbase.com/corp.htm]. June 1998.
27. Wilkinson, S. and others, *Professional JSP*, Wrox Press, 2000.
28. Zikri A., and Dirksen P., *Oracle Designer Generation*, Osborne Oracle Press, 1999.

## INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center ..... 2
   8725 John J. Kingman Road, Ste 0944
   Fort Belvoir, VA 22060-6218

2. Dudley Knox Library ..... 2
   Naval Postgraduate School
   411 Dyer Road
   Monterey, CA 93943-5101

3. Chairman, Code CS ..... 1
   Naval Postgraduate School
   Monterey, CA 93943-5101

4. Professor Thomas Wu ..... 1
   Computer Science Department Code CS
   Naval Postgraduate School
   Monterey, CA 93943-5118

5. Deniz Kuvvetleri Komutanligi ..... 1
   Personel Daire Baskanligi
   Bakanliklar
   Ankara, TURKEY

6. Deniz Kuvvetleri Komutanligi Kutuphanesi ..... 1
   Bakanliklar
   Ankara, TURKEY

7. Deniz Harp Okulu Kutuphanesi ..... 2
   Tuzla
   Istanbul, TURKEY

8. Yazilim Gelistirme Grup Baskanligi ..... 1
   Deniz Harp Okulu Komutanligi
   Tuzla
   Istanbul, TURKEY

9. Envanter Kontrol Merkezi Komutanligi ..... 1
   Golcuk
   Kocaeli, TUZLA

10. Ugur Demiryurek ..... 2
Sirinevler Mah. 3 No lu Sokak
No:2 Kat:3
Sakarya, TURKEY