Web and Database Connection -- Updating a Travel Website to Take Advantage of Oracle and ASP

Baixiang Liu

Follow this and additional works at: https://scholar.dsu.edu/theses
Web and Database Connection
--Updating a Travel Website to Take Advantage of Oracle and ASP

By Baixiang Liu

A Program Project
Submitted in Partial Fulfillment
of the Requirement for the Degree of
Master of Science in Information Systems

Dakota State University
June 28, 2001

Approved by:

Prof. Ronghua Shan, Chair

Prof. Tom Halverson

Prof. Minghua Wang
Web and Database Connection
--Updating a Travel Website to
Take Advantage of Oracle and ASP

By Baixiang Liu

A Program Project
Submitted in Partial Fulfillment
of the Requirement for the Degree of
Master of Science in Information Systems

Dakota State University
June 28, 2001

Approved by:

Prof. Ronghua Shan, Chair

Prof. Tom Halverson

Prof. Minghua Wang
# Table of Contents

I  Introduction.................................................................03

1.1 Project Goals............................................................03
1.2 Project Overview.........................................................05

II Technology Overview....................................................06

2.1 Three Components of Web and Database Connection.........06
2.2 Web and Database Connection Process.............................11
2.3 Web and Database Connection Technologies Overview.....12

III Implementation ..........................................................24

3.1 Implementation Overview.............................................24
3.2 Business Requirement Identification.............................25
3.3 Database Design..........................................................27
3.4 PL/SQL Programming....................................................31
3.5 Oracle Client Setup......................................................33
3.6 PWS Setup.................................................................34
3.7 Web Interface Design....................................................36
3.8 ASP Coding...............................................................50
3.9 Putting Everything Together.........................................54

IV Conclusion ...............................................................55

V Appendix.........................................................................57

A. Sample Codes.............................................................57
B. References.....................................................................101
I Introduction

Currently, many people, especially those working in small and medium-sized businesses, are trying to figure out how to effectively use the Web. In most cases, the primary focus is on using the web to create, manage, find, and deliver information stored in a database. This has led to an explosion in the number of new web-oriented database tools. The goal of this project is to provide an insight of how those products work and how to choose among them. In particular, this project will introduce and compare the most popular Web database technologies, namely, CGI, Java, and ASP. The comparison shows that ASP has some obvious advantages over the others in most of the situations. A complete working example will be implemented, in which fully functional ASP programs are used to connect to Oracle, the most popular relational client/server database in today’s production environment. This example demonstrates functionality on a travel website.

1.1 Project Goals

The web is a great medium for delivering information. Databases are the perfect medium for storing and managing information. Thus a database can be used to manage the “back-end” mechanics of an information delivery system, while the web is used to handle the “front-end” user interface.

This project will first give an overview of the connection process and some popular connection technologies, and then provide a more detailed introduction to one of the specific connection topics—using ASP to connect to Oracle database and to call stored procedures in Oracle database. ASP is one of the most popular and powerful server side technologies and Oracle is one of the most popular relational
client/server databases. However, few people are using ASP to connect to Oracle. There are several reasons for this.

First, ASP is purely Microsoft technology. It has built-in support for Microsoft Access or SQL Server. In almost all ASP books or introductions, whenever they introduce the capabilities of ASP in database connections, they illustrate in detail how to connect to Microsoft Access or SQL Server and stop sharply by saying one can use a similar approach to connect other databases. Secondly, Oracle is one of the major competitors of Microsoft. Oracle is providing a complete set of tools and related technologies by itself to support the Internet such as Oracle Application Server (OAS) and WebDB. The third reason is that IT professionals can be roughly separated as Microsoft professionals and non-Microsoft professionals. Microsoft professionals tend to use Microsoft technologies in whatever they do. They work on Windows operating system; they use VC++ and VB to do programming; they use FrontPage to develop web sites; and they use Microsoft Access and SQL Server when they need databases. Non-Microsoft professionals would just do everything using alternative technologies. Because of the above three reasons, if you search the Internet using keyword “connecting to Oracle using ASP”, few results will pop up. There is big gap between the two and this project will try to narrow this gap in one particular area using Microsoft ASP to connect to Oracle. As you will see, there are some traps and challenges involved when implementing the connection. Calling stored procedures of any database is always an advanced topic in any programming technology. In this project, I called stored procedures to do the data manipulation to achieve better performance using ASP. As you will see, it gave me even more challenges and it also made this project even more worthwhile.
1.2 Project Overview

In Section I, after presenting the project goals and an overview of the project, I start by introducing three components involved in the connection, namely, web browser, web server, and web database. The connection could not be accomplished without all of them. Then, I briefly discuss three of the most popular programming technologies: CGI, Java, and ASP. They are the glue that sticks the three components together. Their background knowledge is provided; their advantages and limitations are addressed and compared; their future development is predicted.

In Section II, a working ASP web application is used to illustrate the whole ASP and Oracle connection process. An existing travel page, without database support, is updated by adding an Online Tour Booking System to make the page more dynamic and interactive. Here is a summary of what is included in the implementation part:

1. I started by designing an Online Tour Booking database by creating and populating several tables, and then I designed PL/SQL package, customer_pkg, with several stored procedures which help to accomplish data manipulation in the database. The tables and package specification and body are all stored in an Oracle database server.

2. I installed and configured Personal Web Server (PWS). I had to do this in order to test my programs.

3. With the database and the web server correctly setup and configured, I designed a web interface to let customers and tour operators interact with the Web. Then I wrote several ASP programs to implement the connection through the PWS. The ASP programs will also call the PL/SQL package to do a Web Server. The ASP programs are stored in the Web Server and acting as a middle tier in the
whole process.

In the implementation section, particular technical difficulties are addressed such as Oracle and PWS configuration problem and ASP connecting string problem.

In Section VI, an outline of the project is given and final conclusion is made. The advantages of database supported web pages are reiterated. Project structure is further clarified.

II Technology Overview

2.1 Three components in database and web connection process

As illustrated in Figure 2-1-1, the web browser, web server, and web database work together to display and store information in the Internet environment. Each of them is responsible for different parts of the job.

![Diagram of three-tier architecture]

Figure 2-1-1: Three tier architecture
1. Web Browser

A web browser is a software program used to access the World Wide Web, the graphical portion of the Internet. Web browsers can recognize the HTML tags and display the documents accordingly. The first browser, called NCSA Mosaic, was developed at the National Center for Supercomputing Applications in the early '90s. The easy-to-use point-and-click interface helped popularize the Web, although few then could imagine the explosive growth that would soon occur.

Although many different browsers are available, Microsoft Internet Explorer and Netscape Navigator are the two most popular ones. Netscape and Microsoft have put so much money into their browsers that the competition can't keep up. The Netcraft Web Server Survey is a survey of web browser and web server software usage on the Internet. According to most recent Netcraft survey conducted in May 2000, IE has about 45% of the market share, while NN has about 42%.

2. Web Servers

Web server is an essential part of client/server structure in the web and database communication. There are a vast number of web servers available. Many are available free of charge such as Apache Server.

Web servers allow you to serve content over the Internet using the Hyper Text Markup Language (HTML). The Web server accepts requests from browsers and then returns the appropriate HTML documents.

HTTP is the protocol that allows web browsers and servers to communicate. It forms the basis of what a web server must do to perform its most basic operations. HTTP started out as a very simple protocol, and even though it has had numerous
enhancements, it is still relatively simple. As with other standard Internet protocols, control information is passed as plain text via a TCP connection.

Web servers are designed around a certain set of basic goals:

- Accept network connections from browsers.
- Retrieve content from files or databases locally or remotely.
- Run local programs stored in web servers.
- Transmit data back to clients.
- Be as fast as possible.

According to Netcraft survey conducted in January 2001, the most popular web servers are Apache (about 59% market share) and Microsoft Internet Information Server (about 21% market share). Microsoft Personal Web Server (PWS) is a slimmed down version of IIS. It is free for download and can be installed on Window95 or Window98. That is why PWS is used to test my programs in this project.

3. Web Database

Putting data onto the World Wide Web in the form of a fully accessible, searchable database can open up a wide variety of possibilities for teaching, learning, marketing, and research. For example, when connected to a simple web page form, a database can be used to gather social research data cheaply and easily via online surveys.

What kind of databases are best choices to support the Web? These days, there are many different kinds of databases. The most popular database today is called relational database. The relational model was developed out of the work done by Dr. E. F. Codd at IBM in the late 1960s.
At the core of the relational model is the concept of a table (also called a relation) in which all data is stored. Each table is made up of records (horizontal rows also known as tuples) and fields (vertical columns also known as attributes). It is important to note that how or where the tables of data are stored makes no difference. Each table is identified by a unique name and that name can be used by the database to find the table behind the scenes. As a user, all you need to know is the table name in order to use it. You do not need to worry about the complexities of how the data is stored on the hard drive.

This data access methodology makes the relational model a lot different from and better than the earlier database models because it is a much simpler model to understand. This is probably the main reason for the popularity of relational database systems today.

Another benefit of the relational system is that it provides extremely useful tools for database administration. Essentially, tables cannot only store actual data but they can also be used as the means for generating meta-data (data about the table and field names which form the database structure, access rights to the database, integrity and data validation rules etc).

Thus everything within the relational model can be stored in tables. This means that many relational systems can use operations recursively in order to provide information about the database. In other words, a user can query information concerning table names, access rights, or some data and the results of these queries would then be presented to the user in the form of a table.

Most databases that you will come across these days will be relational databases. However, there are many types of relational databases and not all of them will be
useful for web applications. In particular, it will be the client/server databases that you will use for the web.

A client/server database works like this: A database server is left running 24 hours a day, and 7 days a week. Thus, the server can handle database requests at any hour. Database requests come in from "clients" who access the database through its command line interface or by connecting to a database socket. Requests are handled as they come in and multiple requests can be handled at one time.

For web applications which must be available for worldwide time zone usage, it is essential to build upon a client/server database which can run all the time.

Why is Oracle used in this project?

As we mentioned above, most of the databases used in the Internet today are client/server relational databases such as SQL Server, Oracle, Informix, Sybase, DB2 and so on. In this project, I used Oracle in particular to implement my online management system. The reason is that Oracle is the most widely used database in the production life to support the Internet especially for big enterprises; Oracle, especially Oracle8i, is designed to support the Internet; it is scalable, secure, and reliable; it supports large transactions; it can store all types of data such as binary, audio/video, and images; it provides easy ways of backup and recovery. Its main limitation is its complexity. Its architecture and background processes are very complicated. But once you have mastered it, its power is virtually unlimited.
2.2 Web and Database Connection Processes

As shown in Figure 2, the following steps are involved in retrieving data from a web database:

1. The web client makes a request using some sort of form or hypertext link (like a button).
2. The request is sent to the web server through HTTP.
3. The web server receives the request and passes it to the middleware through technologies such as CGI, Java Servlets, or ASP.
4. The middleware processes the request, formulates the appropriate SQL
commands, and passes it to the database server using ODBC or to its own
database server as appropriate.

5. The database server receives the SQL request through ODBC and translates it
into native instructions.
6. The database server receives the requested data from the database and sends it
back to the middleware.
7. The middleware formats or processes the returned data into some format
appropriate to the web and sends it to the web server.
8. The web server returns the data from the database to the web client.

2.3 Web-Database Connection Technologies Overview

There are countless web-database connection technologies today. No single project
can cover them all. Here I will just cover three most popular ones: CGI, Java, and
ASP

1. CGI Programming

In the early days of HTML and the Mosaic browser, documents could only be
marked up for web delivery. The interaction between the client and the server was
essentially one-way. It quickly became apparent, even before the commercial world
intruded on the Web, that a way to transfer data from the user to the server for
processing would be valuable. Thus, the Common Gateway Interface (CGI) was
born.
CGI: Facts-at-a-Glance

<table>
<thead>
<tr>
<th>Platforms</th>
<th>Virtually all platforms.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pros</td>
<td>Can use with all web servers.</td>
</tr>
<tr>
<td></td>
<td>Can use any script or programming language on the server for programs.</td>
</tr>
<tr>
<td>Cons</td>
<td>Slow performance.</td>
</tr>
<tr>
<td></td>
<td>Hard-to-write scripts and programs for database access.</td>
</tr>
<tr>
<td>Price</td>
<td>Built into essentially all web servers.</td>
</tr>
<tr>
<td>Extras</td>
<td>Interpreter/compiler for processing the script or program</td>
</tr>
<tr>
<td></td>
<td>External function libraries to cut production time.</td>
</tr>
</tbody>
</table>

CGI is a way for the web server (and thus the web client) to interact with any other software program on the server. Plain HTML is static. CGI provides a way for the user to affect output and thus make the web more dynamic. The web server starts a process on the server by running a script or executing a program. The results of that script or program are then sent back to the web server, which can echo them back to the web client.

CGI applications can take input from HTML forms, the client environment, and the server environment. This means that web forms can serve as the front end for a database program running on the server. In any client/server application, the local client is just a mean to get data into and out of the server, which handles the real work. In other words, the client software is simply an interface to help humans communicate with the server. The web browser is starting to replace proprietary program clients throughout the computer industry, and CGI was the original catalyst that started this movement.

CGI is designed to provide a gateway between web servers and other applications
on the server. It is now part of the HTTP standard, so all commercial servers provide CGI support. Many web servers also have proprietary APIs for communication between the web server and other server applications, but CGI has the advantage of being easier to use and more widely supported.

It's important to remember that there is no such thing as a program written in CGI; CGI programs are written in a scripting or programming language. CGI is the enabling technology that handles the transfer of information between the web server and other applications. There are few limitations as to which languages are used to write CGI programs, but for most people, the phrase “CGI application” implies that it was written in Perl.

Perl is a Unix-based programming language optimized for parsing text. It predates CGI and the Web, but is very well suited for working with HTML form data from users. Since the majority of data on web pages is text, being able to easily manipulate it is important. Text manipulation is what Perl is designed to do, so it's the language of choice for most CGI applications.

CGI programs use HTML forms to accept requests from users, search the databases, and provide customized responses. CGI programs were the forerunners of the new, active web. Simple, stored pages sufficed for information display, but people now used web applications to find those pages. Web site programmers began to use CGI programs to provide dynamic content rather than static HTML.

Unfortunately, CGI programs have several shortcomings. The Web is a world of small transactions. Each transaction consists of a single request or a series of short requests by the browser. When you request a page, the web server supplies the page, and then forgets about you. For busy sites, such requests may happen at the rate of 100 requests per second or more. The first generation of CGI programs
consisted of executables that had to be loaded into memory for each request. The CGI program would then process the request and terminate. All of this loading and unloading used up time and resources on the server. It obviously takes considerably more memory and Central Processing Unit (CPU) time to load a program and execute it than it does to simply return the contents of an HTML file.

Connecting web with database using CGI is quite straightforward. One sample code of CGI-Oracle connection is given below:

```perl
Use DBI;
$dbh = DBI->connect('dbi:Oracle:DB', 'username', 'password');
or
$dbh = DBI->connect("dbi:Oracle:host=$host;sid=$sid", $user, $passwd);
```

The DBI is a database access module for the Perl programming language. It defines a set of methods, variables, and conventions that provide a consistent database interface, independent of the actual database being used. It is important to remember that the DBI is just an interface. The DBI is a layer of “glue” between an application and one or more database driver modules. It is the driver modules which do most of the real work. The DBI provides a standard interface and framework for the drivers to operate within. The DBI “dispatches” the method calls to the appropriate driver for actual execution. The DBI is also responsible for the dynamic loading of drivers, error checking and handling, providing default implementations for methods, and many other non-database specific duties.

Each driver contains implementations of the DBI methods using the private interface functions of the corresponding database engine. Only authors of sophisticated/multi-database applications or generic library functions need be concerned with drivers.
For the above code to work, DBD::ORACLE module must be installed. DBD::Oracle is a Perl module which works with the DBI module to provide access to Oracle databases. CGI is an interface program that is stored on the remote web server and executed on the web server in response to a request from a user. It is one of the earliest technologies that made the world wide web dynamic and interactive. Although, some other similar technologies which came into existence later showed some advantages over CGI, CGI remains to be widely used in the real world today.

2. Java

Java, which was based on the object-oriented C++ programming language, was designed from the beginning to be used for programming applications that run in a distributed network environment. The Internet (or an intranet) exactly fits the description of the environment for which Java was designed. Server-side Java applications include Java Servlets and Java Server Pages (JSP). Servlets are Java technology's answer to CGI programming while JSP is Java technology's answer to Microsoft's ASP. Whenever I mention Java in this project, I am referring server side Java, since this project is mainly dealing with server side technologies.

Facts-at-a-Glance

Platforms: Virtually all platforms.
Pros: Platform-independent, object-oriented programming language for network applications.
Cons: Performance is still slow compared to most other sever side technologies.
Price: Free (Sun JDK).

Java is an excellent tool for developing a wide variety of web applications. It has a number of attractive features for web development, including portability between
any conceivable platform and robust security features for network environments. It also delivers better performance than many web application alternatives. But Java is still a young language that has yet to live up to its full potential; to date, performance can be sluggish and some of the network security features make it difficult to implement certain kinds of applications. The future of Java is bright, however, and it is very useful in a large number of web application development scenarios.

Java objects are typically called classes. They consist of a set of data and methods that operate on that data. Classes can be hierarchical with subclasses inheriting the behavior of the parent (or superclass). This is a feature prevalent in many modern programming languages, such as C++. The Java classes are further grouped into packages, which consist of related objects. Examples of Java packages include the Abstract Windowing Toolkit (AWT) or later Swing, which provides the standard visual components like buttons and scrollbars, and various database connection packages.

Here is a Java-Oracle connection example and brief illustration

```java
import java.sql.*;

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
java.sql.Driver myDriver=
// new sun.jdbc.odbc.JdbcOdbcDriver();
java.sql.DriverManager.registerDriver(myDriver);
String url = "jdbc:odbc:DSN"; //jdbc:<subprotocol>::<subname>[(odbc dsn)
Connection con = DriverManager.getConnection(url, "Fernanda", "J8");
```

...
JDBC, the Java Database Connectivity, is one of the key players in making Java an appropriate language and platform for building world-class application. JDBC API allows Java applets, Servlets, and standalone application to access data stored in relational database management systems in a universal way. Similar to DBI, It is totally independent of the various database management systems. It can also be implemented on top of Open Database Connectivity (ODBC).

Establishing a connection to a database using JDBC involves two steps as shown in above sample code: loading the driver and making the connection using the driver. In the above example, The method, `forName`, defined in `java.sql.*` package is used to load the JdbcOdbcDriver. You got to load different diver for different DBMS.

An appropriate Data Source Name(DSN) must be created before you make the actual connection. The URL string (with the DSN) is used to create connections to the database. For this purpose, the `getConnection()` method on the driver manager is used to request a connection object.

The programming language Java is compiled into an abstract machine code, and hence may be run on any machine. It can be used both in server side and client side. It is a purely object-oriented programming language. It is clearly the most sophisticated language for developing portable web applications. It is the perfect tool for adding dynamic features to web pages and building useful web-based programs for business. There is a lot of work left to make Java live up to its promises, but it is likely to become widely used in the near future and to replace CGI and some other tools for high-end web application development.

3. ASP
Active Server Pages (ASP) use the power of the web server to process user requests and provide dynamic, individualized, content based on logic, file, and database data. In other words, ASP lets multiple users simultaneously run a program on your web server.

ASP Facts-at-a-glance

Platforms  Mainly Windows

Pros  The code inside ASP is mixed in with standard HTML and is never seen by the browser. ASP pages run in all browsers unless the person making the page uses HTML or browser commands outside of the portions.
ASP runs faster than CGI and Java.
Powerful in making Web more interactive

Cons  Limited in Microsoft technologies.

Price  Free and already built into Win2000. It is part of IIS and must be added with add/remove programs menu there.
Free for NT4 or Win 95/98 if one installs the NT4 Option Pack.

There are many benefits to ASP and this project will use ASP programs to demonstrate the real coding in Web and database connections. The followings are a couple of general reasons why ASP is the choice for this project.

ASP is language-independent. The ASP engine does not depend on a single language. In fact, the ASP engine doesn’t actually execute the code you write.
Instead, ASP is a language-independent scripting host. The ASP engine works with any scripting language that is compatible with the Microsoft Scripting Host requirements. It can even work with code written in multiple scripting languages on the same page. The ASP engine differentiates scripting code from HTML, and then asks the appropriate scripting engine(s) to execute it.
ASP is for non-programmers. ASP is designed based on the idea that programmers will create reusable, general-purpose entities (the blocks) and anybody with a few hours to spare can then hook them together to create complex programs.

ASP’s strength lies in providing simple decision-making capability to what would otherwise be static HTML pages, and in coordinating and monitoring back-end components to return quick HTML responses to disparate clients. In other words, ASP is the glue. But it’s not yet simple glue. Although you may not need to be a programmer to build simple ASP pages, you need to become a programmer to build ASP applications.

ASP code resides in text files. Text files are easy to modify, even after deployment. It’s a tremendous advantage to be able to fix a problem remotely using a text editor. Web applications that depend on compiled code or registered ActiveX objects are much more difficult to maintain and upgrade.

ASP code times out. IIS stops executing ASP pages after 90 seconds by default. You can (and normally should) adjust this to a shorter value. Therefore, if you accidentally write an endless loop, or allow someone to request a million records, you won’t tie up the server beyond the timeout interval. Many Internet Service Providers (ISPs) who won’t usually host compiled applications will host ASP applications—partly because the scripts time out.

ASP code is server-safe. ASP code runs in a limited space—for example, you can’t natively read or write binary files with ASP. It’s very difficult, if not impossible, to completely crash an IIS server with native ASP script. That’s another reason many ISPs will host ASP applications when they refuse to host Web applications developed with other technologies.
ASP applications are usually small. Because all the DLLs are already installed on the server, you need only deliver the code files, images, and support files to make an ASP application run—and those files are usually small and highly compressible.

You can upgrade ASP applications without stopping IIS. Although it may not sound like it, this is a major advantage. It's no advantage at all when your application is the only one running on the server, but when there are dozens of applications running on the same server (typical of larger businesses), no one wants to stop or shut down the server to make changes in your application.

ASP-Oracle connection example and brief illustration

```vbnet
Const cConnection = "SERVER=design;driver=\{Microsoft ODBC for Oracle\};UID=liu;PWD=literal;"
Dim objConnection
Set objConnection = Server.CreateObject("ADODB.Connection")
objConnection.ConnectionString = cConnection
objConnection.CursorLocation = 3 /* adUseClient
objConnection.Open
```

Similar to CGI and JDBC, an appropriate driver and data source name must be provided before the connection. Then an ADO connection object can be created. After that, you can use the open method to connect the Oracle database. More information about this is given in the implementation section.

In conclusion, ASP is one of the most popular sever side technologies. It is language-independent and easy to learn. It runs faster than CGI and Java Servelet and its code is server-safe. It is good at making quick decision in processing user inputs.

4. Comparison of CGI, Java, and ASP
Through the above overview of the three most popular web database technologies, you may have noticed some similarities among them. Some of the similarities are: they are all powerful tools and popularly used in the production environment; they share the same connection logics; they need to have an interface between our programs and databases (DBI, JDBC, and ADO); an appropriate driver must be installed or loaded to implement the functions defined in the interface; and enough Information about the database server must be provided in the connection string.

On the other hand, they have some fundamental differences. CGI programs, earliest among the three, run mostly on Unix; CGI programs, especially those written in Perl, are hard to write and to understand; it is also slower than Java and ASP. Java is platform independent; Java code is easier to write and read than CGI, but harder than ASP; the semi-compiled code runs faster than CGI, but slower than ASP. ASP, a Microsoft technology, is Window based; it is the fastest of the three technologies.

Based on their functional differences, there are certain areas where each excels:

Text manipulation, CGI written in Perl is powerful in processing text, it would be the best technology to choose. GUI and multimedia design in the Web, Java would be the best choice. Web dynamic and interactivity, ASP would be the most appropriate technology to make the web more dynamic and more interactive which are the basic for E-commerce.
III Implementation

Having introduced web and database technologies in theory, now it is time to use a real example to demonstrate how to implement ASP and Oracle connection. Here I am going to use ASP, the most popular web-database connectivity technology to connect to Oracle, the most popular web database.

A quite complicated process is involved in an ASP-Oracle connection. I started with identifying business requirement of the travel agency, and designed the database according to the database design life cycle. My second step was to set up Oracle Server and Client architecture and web server. After that, I developed PL/SQL programs and ASP programs. Finally, I put everything together to make my Online Tour Booking System work.

3.1 Implementation Overview

As a demonstration, I have updated an existing travel agency website. The website’s URL is http://www.chinatravelservice.com. The website was well designed and had some nice pictures and introductory text. However, because it was not connected to a database, it was not interactive and dynamic. My friend, Mr. Xiwen Zhang, is the president of the travel agency. He had asked me to update the website to make it more interactive and dynamic. I agreed to implement it into my graduate project.
3.2 Business Requirement Identification

China Travel Service was established in 1928 in China, and its first office in the United States was set up in San Francisco in 1982. Now there are three branch offices over the US and more than a hundred employees. It is one of the oldest and largest tour operators organizing tours to China in the world, and maintains excellent relationships with the hotels and airlines serving the region.

After talking with several related persons including Mike Johnson, the sale manager, and Bob Zhou, the web administrator, I basically outlined their business requirement and critical success factors as followings:

1. To enlarge tourist market share geographically.

Presently, their market is limited in the West Coast where they have about 15% of market share. But due to the intense competition in the area, the market share is shrinking noticeably. Mr. Zhang, the president, concluded in a corporate meeting that “if we want to be sustained, we have to open new customer bases in the US Central and the East Coast.”

2. To increase the percentage of non-Chinese tourists.

Currently, 59% of their customers are Chinese living in overseas. They seldom need full service-packaged tours, because most of them have relatives or friends in China to help with their accommodation. They tend to bargain a lot to save some money for their Chinese relatives. So the net profits per customer are generally less than other non-Chinese customers. Increasing the percentage of non-Chinese customer thus increasing the total number of customers per year becomes a crucial success factor.
3. To use Web, Database and other IT technologies to improve service

Online ticketing and tour booking are among the most active and popular E-commerce activities on the Web. Almost all travel agencies are pushing their business on the Web by utilizing some of the most advanced technologies to improve services and to attract more customers. Getting ahead with the current technology contributes to the profits for the company. The management of China Travel Service has realized this, so they decided to update their website to include an Online Tour Booking system to improve their tour booking process.

4. To have a competitive advantage over their competitors.

This can be done by connecting the database application (Oracle) to the website. Customers’ information will be collected and this will help them to serve their customers specifically to their needs than competitors.

5. To offer 24/7 service

By introducing the Online Tour Booking System on the website, it will give our existing customers and prospective customers around the clock and globe booking services.

6. To develop a good customer relationship management.

This will be done by using the data that is collected from the website entered the customers when they book, which will help them to know the purchasing behavior of their customers and create a good relationship with the customers.

7. To be able to distinguish between the Chinese and the Non-Chinese customers

Using Online Booking System, this can be easily done. Base on this, the
marketing persons can further identify the different purchasing behavior and patterns, so that they can adjust their marketing strategies.

Further communication with Mike and Bob revealed two business problems that need to be solved urgently. The sale department needs more precise and timely information of the market trend, which could be possibly achieved by analyzing geographical and demographical customer information. The IT department requested more advanced IT expertise to implementing an online tour booking system in their existing website. They all believe that the existing website is not interactive and dynamic enough to accommodate their business needs. In order to accomplish those requirements, I suggested to implement the online tour booking system using Oracle as a back-end support and ASP as glue. My suggestion was discussed in a corporate meeting and adapted unanimously, since I had introduced the power of both ASP and Oracle to the management in some details before the meeting.

3.3 Database Design

Immediately after the project was approved, I started to collect data related to the business with the help of Mike. I was especially interested in the information about customers, itinerary, meal, booking, and hotels, since the information is more related to online tour booking. A data model was drawn based on my initial analyses as shown in Figure 3-3-1.
Figure 3-3-1: Initial Data Model

In the initial data model, five data entities and their relationship are identified. The entities are *Booking, Customer, Itinerary, Meal, and Hotel*. Other entities such as employee entity, may also be important to the management, but may not be crucial to the online tour booking system, so they are not included here.

My next step was to identify attributes for the above entities. Based on my several years working experience as a travel agent and the inputs from the related persons of the company. The following attributes of each entity in my preliminary data model were identified.

Customer: *CustID; CustFName; CustLName; CustBDate; CustAddr; CustCity; CustState; CustZip; CustHPhone; CustOPhone; CustFax; CustEmail.*

Itinerary: *ItinID; ItinSPlace; ItinEPlace; ItinDuration; NumOfPlace; ItinQuotation; ItinDesc.*
Booking: BkgID; BkgCode; BkgDate; CustID; ItinID.

Meal: MealID; MealCode; MealDesc.

Hotel: HtlID; HtlName; HtlRate; HtlLocation; HtlPhone; HtlFax; HtlEmail; HtlRep.

I explained to Mike and Bob what each attribute actually means. They were a little confused by the BkgID and BkgCode attributes in the Itinerary entity. “Why do we need two columns which mean almost same?” they asked me. I told them BkgID is the primary key of the Itinerary table. It uniquely identifies each row and will be used mainly by database internally. BkgCode refers to specific tour package. It is more user-friendly code and will be used mostly by customers and tour operators. They basically agreed with me, but suggested some additional data. They told me that when a customer books a tour, he usually has some special preferences about his tour. So I added one more attribute BkgComm in the Booking entity for customers to put their preference information.

Based on above analysis, a simplified final ER diagram was drawn:
Figure 3-3-2: ER Diagram of the Online Tour Booking Database

Figure 3-3-2 is a simplified version of ER diagram of the Online Tour Booking database. Only the primary key attribute of each entity is shown in the diagram. It was approved by the management, since it basically reflected their related business requirement.
Finally, the physical database was created according to the ER diagram and entity and attributes identifications. Some sample data are also inserted into the tables. The SQL scripts of physical table creation and insertion are all listed in the appendix.

3.4 PL/SQL Package Design

In order to achieve better performance in retrieving data from the Oracle server, I created a PL/SQL package, in which several stored procedures and PL/SQL table data types were all defined. A package is a way of grouping collections of objects together, such as data types, stored procedures, functions, variables, and constants. The essential part to remember is that the TYPE declaration for each PL/SQL table must appear in the specification section of the package. A package consists of a specification (header) and a body section.

The specification section is the definition of all elements that can be referenced outside the package, i.e. the public elements. This is where the type PL/SQL table declaration must be placed. The following code are snapshot from my customer_pkg specification:

```
CREATE OR REPLACE PACKAGE customer_pkg
AS
    TYPE tblID IS TABLE OF NUMBER INDEX BY BINARY_INTEGER;
    TYPE tblFirstName IS TABLE OF VARCHAR2(30) INDEX BY BINARY_INTEGER;
    TYPE tblLastName IS TABLE OF VARCHAR2(30) INDEX BY BINARY_INTEGER;
    TYPE tblSalary IS TABLE OF NUMBER INDEX BY BINARY_INTEGER;
    TYPE tblStartDate IS TABLE OF DATE INDEX BY BINARY_INTEGER;

    PROCEDURE GetCustomerList(o_custID OUT tblID,
                                o_FirstName OUT tblFirstName,
                                o_LastName OUT tblLastName);
END customer_PKG;
```
Five variables of table data type and one stored procedure are defined. The hardest part to understand in the above code is the TABLE data type. A PL/SQL table is an indexed, single-dimension, unbounded, sparse collection of homogenous elements. It is a data type similar to arrays as found in VB and other programming languages. It defines a structured (non-scalar) data type capable of storing a single array of values.

The body section then defines the actual code that makes up the objects within the package, i.e. the implementation of the stored procedures. The following codes are snapshot from my customer_pkg body:

```
CREATE OR REPLACE PACKAGE BODY customer_pkg
AS
    PROCEDURE GetCustomerList(o_emID OUT tblID,
                              a_FirstName OUT tblFirstName,
                              o_LastName OUT tblLastName)
    IS
        CURSOR customer_cur IS
            SELECT em_id,
                   em_first_name, em_last_name
            FROM customer;
        recCount NUMBER DEFAULT 0;
    BEGIN
        FOR CustomerRec IN customer_cur LOOP
            recCount:= recCount + 1;

            o_emID(recCount):= CustomerRec.em_id;
            o_FirstName(recCount):= CustomerRec.em_first_name;
            o_LastName(recCount):= CustomerRec.em_last_name;
        END LOOP;

    END GetCustomerList;

END Customer_Pkg;
```

The easiest and most efficient way to populate each of your parameter elements is to use a PL/SQL cursor For...Loop to loop through the results of my cursor SELECT transferring them into the corresponding positions; for each record found increment a
record counter and transfer the first name into o_custFirstName(recCount) and last name into o_Last_name(recCount).

Once defined, the customer_pkg can be compiled and stored in the Oracle server, ready to be called by my ASP code. But before that, I had to setup the Oracle Client software correctly to connect to an Oracle Server.

3.5 Oracle Client Setup

The first and most important thing to remember is that we must install the Oracle client on the same computer where web server is located. Otherwise, the connection of the web-Oracle connection can never be successful. This is the first trap I fell in when I started this project.

The IP address of the Oracle server must be known before making the connection. The installation of Oracle client software is quite straightforward. Just remember to install it on the web server and follow the step-by-step instructions. After the installation, there are two ways to configure it to connect to the Oracle server: edit the tnsnames.ora file or use Net8 Easy Config. The tnsnames.ora can usually be found in $ORACLE_HOME/network/admin directory. The destination and name of the Oracle server should be defined in this file. Here is a sample of tnsnames.ora file I used in the connection.

CMEXAMPLE.WORLD =
(DESCRIPTION =
(ADDRESS_LIST =
 (ADDRESS = (PROTOCOL = tcp)(PORT = 1610)(HOST = CM_SERVER))
 (ADDRESS = (PROTOCOL = tcp)(PORT = 1521)(HOST = LSNR_SERVER))
)
(CONNECT_DATA = (SID = ORCL))
(SOURCE_ROUTE = yes)
)
INFS484.WORLD =
(DESCRIPTION =
(ADDRESS = (PROTOCOL = TCP)(HOST = 138.247.64.201)(PORT = 1521))
(CONNECT_DATA = (SID = ORCL))
)

DESIGN.WORLD =
(DESCRIPTION =
(ADDRESS = (PROTOCOL = TCP)(HOST = 138.247.64.201)(PORT = 1521))
(CONNECT_DATA = (SID = d704))
)

In the above example, I defined an alias "design" which is the name I am going to use in the ASP code to refer to the Oracle server. The Protocol I used is TCP. The Oracle server’s IP address is 127.0.0.1 (local host) and the port number is 1521 (default for Oracle). Oracle instance name (SID) must also be defined. The alternative way is to use Net8 Easy Config. It is a GUI tool used to edit the tnsnames.ora file in the same way as the manual approach. I just went to Start - >Program->Oracle for Window->Net8 Easy Config and followed the instruction to give all the information mentioned above. In order to test the connection after the manual configuration, I started SQL Plus and provided my user name, password, and the connection string "design" which I defined during the configuration. The popup SQL Plus prompt showed the connection between Oracle client and server was successful!

3.6 Personal Web Server Setup

In order to test my project, I used Personal Web Server (PWS) in this project, because it is free for download and I can get full control of it and configure it at
will. PWS is Microsoft's slimmed-down web server, which provides a basis on
which to develop corporate networked application. It can be downloaded free from
Microsoft website. The download and installation process are quite straightforward.
Once installed, PWS runs as small icon in my task tray. By double clicking on this
icon, I started an application called Personal Web Manager, which I used to
administer my web pages. By default, PWS takes the name of my web server from
the name of my computer ("mycomputer"). The name appeared on the publishing
panel of Personal Web Manager.

In order to let users to have access to my ASP application, I added an alias,
ASPCod e, which is known as a virtual directory. Virtual directories are used for
directory paths on the server, where I could store my ASP application. Virtual
directory enables me to construct my web site independently of the directory
structure on my hard drive, and hence disconnect the layout of my web site from
physical layout of the file system. This allows me to move files on my disk
between different folders and drives without having to change the structure of my
web pages. The following steps are involved in creating the alias ASPCode:

1. Start Windows Explorer and create a new directory named ASPCode under
   \inetpub\wwwroot directory created by PWS on the hard drive.
2. Start up Personal Web Manager and click on the Advanced icon.
3. Select the directory <Home>.
4. Click on the add button, and click on the Browse button and select the directory
   \inetpub\wwwroot\ASPCod e that I created in step 1. Add ASPcode to the alias
   box.
5. Make sure the Read and Scripts checkboxes are checked, and that the Execute
   checkbox is empty.
6. Click on OK –the ASPcode appears on the directory tree.
Now, I have created a directory in which to store all of my ASP codes. The ASP codes can be accessed on the Personal Web Server using the URL

http://mycomputer/ASPcode. By that URL, PWS knows that it stands for the directory path c:\inetpub\wwwroot\ASPcode. When executing ASP programs, the URL must call the page via the alias. To test the PWS, simply create an ASP file called ServerTest.asp, store it in the C:\inetpub\wwwroot\ASPcode, and then start a browser and enter the URL http://mycomoputer/ASPcode/ServerTest.asp. If the ServerTest.asp exists and is valid ASP code, the result should be returned to the browser.

3.7 Web Interface Design

The management of China Travel Service has clearly stated that they would not like to see big changes on the web interface, since some of their customers are already used to its layout of retrieving related information from the web site. As I pointed out before, generally speaking, the original interfaces are well designed and also working fine. The problem of the website is not of bad interface design, but of lack of interactivity with users. Here is how the original home page looks:
Figure 3-7-1: Original Home Page

I added a link of Online Tour Booking to an outstanding position just below the name of the travel agency. Other two links were added to the navigation bar. They are an additional Online Tour Booking and Tour Admin. Tour Admin is password-protected page which will link to a page for tour operators to handle those online bookings. The modified home page looks like this:
Figure 3-7-2: Modified Homepage

When either of the Online Tour Booking links is clicked, onlineBooking.asp page will pop up. This is the major page of the Online Tour Booking:
Figure 3-7-3: Online Tour Booking page

If a new user wants to book a tour, he shall register in the database first by clicking the link called New User Sign-up. The newUser.asp program will be called. Here is the New User Registration page:
Figure 3-7-4: New User Registration Page
You may have noticed that there is no input text box for a user to put in his customer ID information. The ID will be automatically entered into the customer table as default by calling custID_sequence.NEXTVAL. By doing so, there are two advantages: 1) duplicate IDs can be eliminated; 2) users may not know what should be entering into the box. The script of creating the custID_sequence sequence is provided in the Appendix A. The NewUser.asp ASP program is responsible for inserting the input user information into customer table when submit button is clicked. The ASP code is also included in the Appendix A.

For existing users, those who are already entered into the customer table, they can just enter their custid and bkgid if they remember that information. Otherwise, they can click on corresponding lookup links immediately followed the input boxes to easily check that information. Each of the lookup links will link to an ASP program which, in turn, will select required information from related database tables. When Check Customer ID link is clicked, the following page will pop:
Figure 3-7-5: Check Customer Id Page

Since one’s email address is unique and easy to remember, I choose email address as search condition to search for the user’s custID. When an email is entered into the email text box, CheckID.asp is called to search the customer table in the database. If the information is found, the custID is returned, otherwise, a message indicating the user has not registered is displayed.
In similar ways, you can look up booking code and itinerary ID by clicking appropriate links in the Online Tour Booking page.

Payment information is also required, but since a third party software need to be purchased, so it is not implemented in this project right now. It will be implemented when the Online Tour Booking system is actually put in use.

When all the booking information are provided and submit button is clicked, Tour Booking Confirmation page will pop up like this:
Figure 3-7-7: Tour Booking Confirmation Page

The Tour Admin link in the modified homepage is designed for tour operators in the travel agency to manage the online tour bookings. One must be an authorized user to login in this page, that is why it is password protected.
Figure 3-7-8: Admin Login Page

A tour operator can view bookings details which have been submitted within a day, two days, a week, or a month. The booking information is selected from booking table by RetrieveBooking.asp ASP program. Tour operators will be provided with following options:
Figure 3-7-9: Retrieve Booking Information Page

One week’s booking information will be retrieved from booking table by
RetrieBeBooking.asp which is called when above information is submitted. The
information is then displayed in a well-formatted table:
Tour operators may need more detailed information about customer, bkgCode, and itinID. Each cell, except that of BkgID, in the table is linked to an ASP program which will retrieve detail information from related database tables. For example, if customer ID 00007 is clicked, RetrieveCustId.asp program will be called and information of Customer with ID 00007 will be displayed like this:
First Name: Baixiang
Last Name: Liu
Date of Birth: Jan 1, 1900
SSN: 234-56-2344
Address: 1051 N. Summit Ave.
City: Madison
State: SD
ZipCode: 57042
Phone: 605-256-6540
Email: liub@pluto.dsu.edu

Back to Admin

California Seller of Travel #1009907-40

Figure 3-7-11: Customer Information Page
Totally, there are over 30 pages designed to implement the Online Tour Booking System. I cannot show them all in the report. But through the sample snapshots, we can see that the Online Tour Booking system is functioning properly.

Following flow chart helps to understand the tour booking process as whole:
3.8 ASP Coding

In theory, it is easy to connect to Oracle using ASP. You just provide proper DSN, userid and password to an ASP connection object like we do to Microsoft Access or SQL Server. The truth is that it is not that simple. I failed so many times while trying to connect to the Oracle database through ASP that I almost had given up this project. When I did search in the Web, there are few materials talking about this and even if I did catch one or two after great efforts, the instructions are usually very vague.

At the beginning, I just tried to install the Microsoft ODBC Driver for Oracle and then create the DSN through the Control Panel in client workstation. It should have worked according to the tutorials. But it didn’t. What I had to do was to do something about the web server. It was not enough for me just to install Oracle driver for ODBC and then create DSN. I had to install Oracle client software and Oracle driver on the Web server as well.

I assumed that the Oracle client software and PWS had been successfully setup according to the instruction in the previous sections. If so, you can test my ASP programs included in the Appendix before I explain some crucial parts of the codes. First, let’s have look at the ASP connection function:

```vbscript
Private Function GetDBConnection
    Const cConnection = "SERVER=design;driver={Microsoft ODBC for Oracle};UID=liu;PWD=litera;"
    Dim objConnection
    Set objConnection = Server.CreateObject("ADODB.Connection")
    objConnection.ConnectionString = cConnection
    objConnection.CursorLocation = 3 'adUseClient
    objConnection.Open
    'Return the new connection back to the calling routine
```

49
Set GetDBConnection = objConnection
End Function

A connection string must be defined first. The most confusing part is what value
should be given to the “Server” parameter here. I tried the IP address of the Oracle
server, it did not work. And then I tried the name of the Oracle server, it didn’t
work either. There are many other names related to Oracle such as Oracle Instance
Name (SID) and Oracle schema. It is really hard to guess that it should be the name
that I myself created as an alias. Remember that “design” is the alias I created
when I did the Oracle client software configuration. Besides that, an appropriate
driver must be provided. In my case, I used Microsoft ODBC for Oracle from
Microsoft Corporation. I could have used other drivers such as Oracle ODBC
driver from Oracle Cooperation. User id and password must also be provided for an
authorized connection. An ADO connection object is created and two properties of
the connection object, connectionString and CursorLocation are defined. With
those properties defined, I called the open function of the connection object and
return an active connection.

Now it is necessary for me to explain how to call the Oracle stored procedures with
ASP. I believe that two keys to understanding how to get recordsets form Oracle
packages are 1) knowing how Oracle returns multiple-row type data, and 2)
knowing the call syntax within ASP.

Oracle does not return recordsets like most other databases do. It returns a multi-
dimensional array of values which ASP can interpret as a recordset. To make
matters more confusing, Oracle does not use “arrays” in its terminology. Instead it
uses “tables”. As I explain before, syntactically, PL/SQL tables are like arrays.
Their implementation, however, is more like a normal database table with two
columns, KEY and VALUE. The type of key is always BINARY_INTEGER and
the type of value is whatever you specify in the definition.

In the ASP code below you will see the syntax for calling the PL/SQL package.

\{call customer_Pkg.GetCustomerList\(\{resultset 100, o_custID, o_FirstName, o_LastName\}\)\}

The general form is:

\{call PackageName.ProcedureName\(\{resultset <NumberOfRows>, Table(array), table(arrays), ....\}\)\}

The "Resultset <NumberOfRows>" indicates the number of rows you expect to be returned. Of course, you usually do not know how many rows to expect. Unfortunately, specifying the number of rows you expect to receive is a required element in a call to an Oracle stored procedure of this type. Fortunately, as long as you specify more rows than you really expect you will be OK. There is no penalty for specifying a large number as far as I can tell.

A standard ADO recordset object can't be used alone to retrieve the data. I used the Command object's Parameters collection to define the arguments to our stored procedure. However, I transferred the results into an ADO Recordset using the Command object's Execute() function in the following way:

```vbscript
Set objCustomerCommand = Server.CreateObject("ADODB.Command")
objCustomerCommand.ActiveConnection = GetDBConnection()
objCustomerCommand.CommandText = cProcName
objCustomerCommand.CommandType = 1    '** adCmdText
Set objCustomerResultSet = objEmployeeCommand.Execute()
```

Now, the retrieved data is stored in the objCustomerResultSet, I can manipulate them at the normal way.
I will not go to every detail of my ASP code. Here is outline of few points that must be remembered when executing the Oracle stored procedure from ADO:

1. For the CommandText SQL statement, wrap {CALL} around the procedure name i.e. {call package_name.procedure_name (params)}

2. Wrap each of the PL/SQL table output parameters with {resultset max_records, paramname}. Here's the most important part, you MUST ensure that the number of records returned does not exceed the number specified, otherwise, you'll receive Oracle error ORA-06512. It is best to limit the number of records transferred by checking returned recordset.

3. The number does not exceed this threshold from within the cursor For..Loop. Each argument needs to be defined separately to return different number of records. Now for the second most important part: the name and order of each parameter to the stored procedure must exactly match those in your calling SQL statement, otherwise you will receive the ODBC driver error "ResultSet column must be a formal argument".

4. Use the standard "?” to define the position of any input parameters. Each “?” represents an element in the Command object’s Parameters collection.

5. Set the CommandText property using the above rules to define your SQL call.

6. Set CommandType to adCmdText.

7. Set the Direction property of any input parameters to adParamInput and set the Value accordingly. For example:

   objCustomerCommand.Parameters(1).Direction = 1 ' adParamInput
   objCustomerCommand.Parameters(1).Value = Request.Form("txtSalary")

With all this done I can then call the Command object’s Execute() function returning the result to an ADO recordset and process it in the normal way.
3.9 Putting Everything Together

The following figure helps us to have a general picture of the implementation process of this project.

Figure 3-9-1: Implementation Process

As shown in Figure 4, three pairs of back and forth communication are involved in the whole process.

1. Web browser → PWS → Web browser

After PWS has been correctly configured, the implementation of this pair of communication is automatically completed based on the default HTTP protocol. Nothing else needs to be done about it.

2. Oracle client → Oracle server → Oracle client

Similar to the web browser and PWS communication, once the tnsnames.ora file is correctly edited to include the Oracle server’s information, there will be no much difficulties to retrieve and send data from the Oracle server via the Oracle client.
TCP Protocol is used in this pair of communication.

3. PWS → Oracle client → PWS

Here comes the most difficult part. The connection is not automatic even if PWS and Oracle client are configured correctly. ASP programs are crucial here to establish the connection. This is the major portion of my project implementation. The crucial points in the connection process are the communication among PWS, Oracle client, and ASP programs. The three of them all reside in a same computer and coordinate with each other to complete required data transmission.

IV Conclusion

Web and database connection is hot topic. There are many technologies that can be used to establish the connection. Most of them are straightforward or being illustrated and introduced in many books or on the Internet. As I illustrated in the Introduction Section, using Java and CGI to connect to Oracle is not very complicated. If you follow some tutorials step-by-step, the connections will be established. Detailed explanations using ASP to connect to Microsoft Access and SQL Server can also be easily found in a book or on the Internet. ASP to Oracle connection, on the other hand, is not introduced very clearly at all. There are two main reasons for this phenomenon: ASP is purely Microsoft technology and it has built-in support for Microsoft SQL Server or Access database; Oracle is one of the major competitors of Microsoft and also has its own integrated web application support. After giving a general overview of web and database connection, this project completed a workable implementation on using ASP to call Oracle PL/SQL package to update existing web page of a travel agency. An Online Tour Booking
System clearly shows the advantages of the web pages with database support over those without. After adding the Online Tour Booking System, the web site becomes more dynamic, more interactive, and therefore more effective. It will surely help the travel agency to improve services, attract more customers, and make more money.

From a technical perspective, a detailed process of web and database connection is illustrated and some technical ambiguities are emphasized. Planning Oracle Client/Server architecture is the basis of database setup. Web Server configuration is crucial to implement database support to the Web. ASP works as glue to stick the web and database together. But using ASP to call Oracle Stored Procedures cannot be done easily. Some technical problems must be solved and correct calling syntax must be followed.

Database design has been emphasized in this project implementation. I followed a database design procedure recommended by some experts. I started the design with business requirement identification, followed by logical design and prototyping, and then by ER Diagramming and physical design. Finally, physical database tables are created and sample data inserted using SQL language.

The Online Tour Booking System implemented in this project serves as a good example showing the effectiveness of the web with database support in general, and it also shows the power of ASP in its quick decision-making and capabilities of Oracle as Web Client/Server relational database in particular.
V Appendix

A. Sample Code

*Name: Create table and insert data scripts
*Designed by: Boaixiang Liu
*Purpose: create database tables and insert into some sample data

CREATE TABLE customer

    ( 
    custID NUMBER(5),
    CONSTRAINT customer_custID_pk PRIMARY KEY,
    custFName VARCHAR2(30),
    custLName VARCHAR2(30),
    custBDate DATE,
    custSSN NUMBER(10),
    custAddr VARCHAR2(50),
    custCity VARCHAR2(20),
    custState VARCHAR2(20),
    custZip CHAR(9),
    custPhone CHAR(10),
    custEmail VARCHAR2(50)
    );

/

INSERT INTO customer

VALUES (custID_sequence.NEXTVAL, 'Liu', 'Tham2', '2 SEP 1994', '134435323', '1014 N Summit Ave', 'Madison', 'WI', 57042, '6082566549',
        'liah@pluto.daw.edu');


```
INSERT INTO customer
VALUES (custID_sequence.NEXTVAL, 'Hodgson', 'James', 2 OCT 1897, 211545399, '43 S Summit Ave', 'Madison', 'WI', 57042, '6052566981', jhh@plazo.dsu.edu);

INSERT INTO customer
VALUES (custID_sequence.NEXTVAL, 'Tom', 'Jean', 8 SEP 1994, 211445313, '3034 N Summit Ave', 'Madison', 'WI', 57042, '6052566949', DSWb@plazo.dsu.edu);

INSERT INTO employee
VALUES (custID_sequence.NEXTVAL, 'Bob', 'Smith', 2 SEP 1995, 211544377, '99 N Summit Ave', 'Madison', 'WI', 57042, '6052561111', pop@plazo.dsu.edu);

INSERT INTO employee
VALUES (custID_sequence.NEXTVAL, 'Sharon', 'Smith', 12 JAN 1990, 211445320, '12 S Summit Ave', 'Madison', 'WI', 57042, '6052560000', smITH@plazo.dsu.edu);

INSERT INTO employee
VALUES (custID_sequence.NEXTVAL, 'Jennifer', 'Brown', 9 SEP 1990, 2113449999, '1022 N Summit Ave', 'Madison', 'WI', 57042, '6052566541', jen@plazo.dsu.edu);

COMMIT;

/

CREATE TABLE itinerary
(id INT IDENTITY(1,1) PRIMARY KEY,
     startPlace VARCHAR(30),
     endPlace VARCHAR(30),
     NumOfPlace NUMBER(6),
     NumOfDay NUMBER(6),
     tour VARCHAR(100));

INSERT INTO itinerary
VALUES 1, 'Beijing', 'Shanghai', 10, 5, '1200.00', 'Jewels of China';
INSERT INTO itinerary
VALUES 2, 'Beijing', 'Shanghai', 15, 10, '2200.00', 'China Local Flavors Tour';
INSERT INTO itinerary
VALUES 3, 'Beijing', 'Hong Kong', 11, 4, '1500.00', 'China Highlights';
INSERT INTO itinerary
VALUES 4, 'Shanghai', 'Hong Kong', 14, 8, '2500.00', 'Culinary & Cultural Tour';
INSERT INTO itinerary
VALUES 5, 'Beijing', 'Hong Kong', 13, 5, '1800.00', 'Golden China';
INSERT INTO itinerary
VALUES 6, 'Shanghai', 'Beijing', 15, 8, '2800.00', 'Ancient Silk Road';
INSERT INTO itinerary
VALUES 7, 'Guangzhou', 'Shanghai', 15, 8, '2600.00', 'Tibetan Adventure');
```
CREATE TABLE booking

  (bookingID NUMBER(5))

  CONSTRAINTS booking_bookingID_pk PRIMARY KEY,
bookingCode CHAR(7) UNIQUE,
bookingDate DATE DEFAULT SYSDATE,
custID NUMBER(5)

  CONSTRAINTS booking_custID_ FOREIGN KEY REFERENCE
customer(custID),
innID NUMBER(5)

  CONSTRAINTS booking_innID_ FOREIGN KEY REFERENCE
itinerary(innID));

INSERT INTO booking

  (bookingID, bookingCode, custID, innID)
VALUES (001, 'JOC-001', 00002, 3);

INSERT INTO booking

  (bookingID, bookingCode, custID, innID)
VALUES (002, 'JOC-002', 00001, 4);

INSERT INTO booking

  (bookingID, bookingCode, custID, innID)
VALUES (003, 'CHT-003', 00005, 2);

INSERT INTO booking

  (bookingID, bookingCode, custID, innID)
VALUES (004, 'GCT-004', 00005, 1);

INSERT INTO booking

  (bookingID, bookingCode, custID, innID)
VALUES (005, 'LETO01', 00004, 2);

CREATE TABLE meal

  (mealID NUMBER(5)) PRIMARY KEY,
mealCode CHAR(7) UNIQUE,
mealDesc VARCHAR2(50));

INSERT INTO meal

VALUES(1, 'AB', 'American Breakfast');

INSERT INTO meal

VALUES(2, 'L', 'Lunch');

INSERT INTO meal

VALUES(3, 'D', 'Dinner');

INSERT INTO meal

VALUES(4, 'SD', 'Special Dinner');

CREATE TABLE hotel

  (hotelID NUMBER(5)) PRIMARY KEY,
hotelName VARCHAR2(30),
hotelRating NUMBER(5),
hotelRate NUMBER(5),
hotelLocation VARCHAR2(30),
hotelPhone VARCHAR2(15),
      hotelID; hotelName; hotelRate; hotelLocation; hotelPhone; custRep;
CREATE SEQUENCE custID_sequence
START WITH 00001
NOMAXVALUE
NOCACHE;

CREATE OR REPLACE PACKAGE customer_PKG
AS
PROCEDURE GetCustList (
    o_custID OUT t_cemID,
    o_FirstName OUT tFirstName,
    o_LastName OUT tLastName)
PROCEDURE AddCustomer (t_custID IN customer.custIDTYPE,
PROCEDURE AddBooking:
   l_custID IN booking_custID%TYPE,
   l_ID IN booking_bookID%TYPE,
   l_bkgCode IN booking_bkgCode%TYPE,

PROCEDURE GetNewBooking:
   o_bkgID OUT booking_bkgID%TYPE,
   o_bkgCode OUT booking_bkgCode%TYPE,
   o_custID OUT booking_custID%TYPE,
   o_billID OUT booking_billID%TYPE,

PROCEDURE GetCustomerDetails:
   o_custID OUT customer_custID%TYPE,
   i_FirstName IN customer_custFName%TYPE,
   i_LastName IN customer_custLName%TYPE,
   o_FirstName OUT customer_FName%TYPE,
   o_LastName OUT customer_LName%TYPE,
   o_dob OUT customer_BDate%TYPE,
   o_ssn OUT customer_custSSN%TYPE,
   o_add OUT customer_custAddr%TYPE,
   o_city OUT customer_custCity%TYPE,
   o_state OUT customer_custState%TYPE,
   o_zip_code OUT customer_custZip%TYPE,
   o_phone OUT customer_custPhone%TYPE,
   o_email OUT customer_custEmail%TYPE

PROCEDURE DeleteCustomer:
   i_FirstName IN customer_custFName%TYPE,
   i_LastName IN customer_custLName%TYPE

PROCEDURE CheckID:
l_email IN customer.email%TYPE,
O_CustID OUT Customer.custID%TYPE)
end customer_pkg;

******************************************************************************
*Name: create customer_pkg body script
*Designed by: Baixiang Liu
*Purpose: implement the procedures defined in the specification
******************************************************************************

CREATE OR REPLACE PACKAGE BODY customer_pkg
AS

******************************************************************************
* Retrive customer list using standard arguments
******************************************************************************

PROCEDURE GetCustomerList(
    O_custID OUT dbEMID,
    O_FirstName OUT dbFirstName,
    O_LastName OUT dbLastName)
IS
    CURSOR customer_cur IS
        SELECT custID,
        CustName, custName
        FROM customer;
        recCount NUMBER DEFAULT 0;

        BEGIN
            FOR customerRec IN customer_cur LOOP
                recCount = recCount + 1;
                O_custID(recCount) := customerRec.custID;
                O_FirstName(recCount) := customerRec.custName;
                O_LastName(recCount) := customerRec.custName;
            END LOOP;
        END;

        END GetCustomerList;
* Add a new customer details using standard arguments

******************************************************************************

PROCEDURE AddCustomer

    i_custID     IN     customer.custID%TYPE,
    i_FirstName  IN     customer.custFName%TYPE,
    i_LastName   IN     customer.custLName%TYPE,
    i_dob        IN     customer.custBDate%TYPE,
    i_ssn        IN     customer.custSSN%TYPE,
    i_add        IN     customer.custAddr%TYPE,
    i_city       IN     customer.custCity%TYPE,
    i_state      IN     customer.custState%TYPE,
    i_zip_code   IN     customer.custZip%TYPE,
    i_phone      IN     customer.custPhone%TYPE,
    i_email      IN     customer.custEmail%TYPE)

IS

BEGIN

    INSERT INTO customer

    VALUES (i_custID,
              i_FirstName,
              i_LastName,
              i_dob,
              i_ssn,
              i_add,
              i_city,
              i_state,
              i_zip_code,
              i_phone,
              i_email):

END AddCustomer;

******************************************************************************

* Add new booking information into booking table

******************************************************************************

PROCEDURE AddBooking (

    i_custID     IN     booking.custID%TYPE),
    i_ID         IN     booking.bookID%TYPE,
    i_bkgCode    IN     booking.bkgCode%TYPE)
IS
BEGIN

INSERT INTO booking VALUES
(l_custID, l_ID, l_bkgCode);

END;

PROCEDURE GetNewBooking (  
  o_bkgID OUT booking.bkgID%TYPE,
  o_bkgCode OUT booking.bkgCode%TYPE,
  o_custID OUT booking.custID%TYPE,
  o_inID OUT booking.inID%TYPE)
IS
BEGIN

SELECT bkgID, bkgCode, custID, inID
INTO o_bkgID, o_bkgCode, o_custID, o_inID
FROM booking;

END;

******************************************************************************
* Retrieves a single customer details using standard arguments
******************************************************************************

PROCEDURE GetCustomerDetails(  
  o_custID OUT customer.custID%TYPE,
  i_FirstName IN customer.custFName%TYPE,
  i_LastName IN customer.custLName%TYPE,
  o_FirstName OUT customer.custFName%TYPE,
  o_LastName OUT customer.custLName%TYPE,
  o_bday OUT customer.custBDate%TYPE,
  o_ssn OUT customer.custSSN%TYPE,
  o_add OUT customer.custAddr%TYPE,
  o_city OUT customer.custCity%TYPE,
  o_state OUT customer.custState%TYPE,
  o_zip_code OUT customer.custZip%TYPE,
  o_phone OUT customer.custPhone%TYPE,
  o_email OUT customer.custEmail%TYPE)
IS
BEGIN

SELECT custID,
       custName,
       custLName,
custID,  
custSsn,  
custAddr,  
custCity,  
custState,  
custZip,  
custPhone,  
custEmail,  
INTO o_custID,  
o_FirstName,  
o_LastName,  
o_dob,  
o_sin,  
o_add,  
o_city,  
o_state,  
o_zip_code,  
o_phone,  
o_email  
FROM customer  
WHERE custFirstName = i_FirstName  
AND custLastName = i_LastName;

END GetCustomerDetails;

(/^********************************************************************  
  * Delete a single customer using standard arguments  
********************************************************************/)

PROCEDURE DeleteCustomer  
   (i_FirstName IN customer.custFirstName%TYPE,  
i_LastName IN customer.custLastName%TYPE)  
IS  
BEGIN  
DELETE FROM customer  
WHERE custFirstName = i_FirstName  
AND custLastName = i_LastName;  
END DeleteCustomer;

PROCEDURE CheckID (  
i_email IN customer.email%TYPE,  
o_CustID OUT Customer.custID%TYPE)
SELECT custID
Into VAR:ESTid
FROM customer
WHERE email = @L_email;
END checked;

END customer_PKG;
/

***********************************************************************

*Name: Home page of modified version

*Modified by: Boiqiang Liu

*Purpose: Add two links to implement the Online Tour Booking System

***********************************************************************

<script type="text/javascript">
  function MM_swapImage() { var i,j=0,666; for(;i=0;i<=6;i++) {
    document.MM_class[0].a[i].src=MM_class[0].a[i].a.src;
  }

  function MM_preloadImages() { if (j="") j=MM_preloadImages.Products; 
    for(i=0;i<=j.length;i++) {
      if((j[i].indexOf(""))+0) j[i].d.MM.i[0].d.MM.p[j[i]]=new Image; d.MM.p[j[i]+x.src=a[i].j]]
    }

  function MM_findObj(n, d) { if (d=document) {if (n=nIndexOf("")) 0; a=frames_frames[1]; return x;
  }

  function MM_swapImage(n, d) {if(d=document) {if(n=nIndexOf("")) 0; a=frames_frames[1]; return x;

  var i, l=MM_swapImage.arguments; document.MM_sr=new Array; for(i=0;i< stavim.he.length;i++) {
    if((i=MM_findObj(i))=null) document.MM_sr[i][0]=0;
    MM_sr[i].x=x.src=a[i]++j[2]++
    }

  var i=0; l=MM_swapImage.arguments; document.MM_sr=new Array; for(i=0;i< stavim.he.length;i++) {
    if((i=MM_findObj(i))=null) document.MM_sr[i][0]=0;
    MM_sr[i].x=x.src=a[i]++j[2]++
    }

  <SCRIPT>
  <LINK href="CTS Home Page_file/cts.css" rel="stylesheet">
  <STYLE type="text/css">.idk2 {
    FONT-WEIGHT: bold; FONT-SIZE: 13px; COLOR: #ff00; LINE-HEIGHT: 14px; FONT-FAMILY: Verdana, Arial, Helvetica, sans-serif
  }

  65
<table>
<thead>
<tr>
<th>Cell 1</th>
<th>Cell 2</th>
<th>Cell 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value 1</td>
<td>Value 2</td>
<td>Value 3</td>
</tr>
<tr>
<td>Value 4</td>
<td>Value 5</td>
<td>Value 6</td>
</tr>
<tr>
<td>Value 7</td>
<td>Value 8</td>
<td>Value 9</td>
</tr>
<tr>
<td>Value 10</td>
<td>Value 11</td>
<td>Value 12</td>
</tr>
</tbody>
</table>

**Table Footnotes:***
- Footnote 1
- Footnote 2
- Footnote 3

**Table Notes:**
- Note 1
- Note 2
- Note 3
<TR>
<TD width='166'>
<CENTER><A href='http://www.uota.com'>
<IMG height='51'
hspace='10' src='CTS Home Page_files/UOTA.gif' width='70'
border='0' &gt;&lt;/A&gt;&lt;/CENTER&gt;&lt;/TD&gt;
<TD width='167'>
<CENTER><A href='http://www.lata.org/'>
<IMG height='51'
hspace='10' src='CTS Home Page_files/lATA.gif' width='70'
border='0' &gt;&lt;/A&gt;&lt;/CENTER&gt;&lt;/TD&gt;
<TD width='166'>
<CENTER><A href='http://www.pasa.org/'>
<IMG height='57'
hspace='10' src='CTS Home Page_files/PASA.gif' width='55'
border='0' &gt;&lt;/A&gt;&lt;/CENTER&gt;&lt;/TD&gt;&lt;/TR&gt;
<TR>
<TD width='166'>
<CENTER><A href='http://www.airchina.com.cn/'>
<IMG height='69'
hspace='10' src='CTS Home Page_files/AirChina.gif' width='75'
border='0' &gt;&lt;/A&gt;&lt;/CENTER&gt;&lt;/TD&gt;
<TD width='167'>
<CENTER><A href='http://203.127.117.110/'>
<IMG height='65'
hspace='10' src='CTS Home Page_files/cashtel.png' width='100'
border='0' &gt;&lt;/A&gt;&lt;/CENTER&gt;&lt;/TD&gt;
<TD width='166'>
<CENTER><A href='http://www.uai.com/'>
<IMG height='14'
hspace='10' src='CTS Home Page_files/UAI.gif' width='150'
border='0' &gt;&lt;/A&gt;&lt;/CENTER&gt;&lt;/TD&gt;&lt;/TR&gt;&lt;/TABLE&gt;&lt;/BODY&gt;&lt;/HTML&gt;

---

*Name: addBooking.html

*Written by: Baixiang Liu

*Purpose: Create booking form for a customer to input data

---
Enter your booking information here. If you forgot your ID or other booking information, Please click the link right after each input box for related information.

<p style="margin-top: 2; margin-bottom: 2; align: left"><font face="Times New Roman" size="3">CUSTOMER ID</font><br/>
</p>

<p style="margin-top: 2; margin-bottom: 2; align: left"><font face="Times New Roman" size="3">BOOKING CODE</font><br/>
</p>

<p style="margin-top: 2; margin-bottom: 2; align: left"><font face="Times New Roman" size="3">Check Customer ID</font><br/>
</p>

<p style="margin-top: 2; margin-bottom: 2; align: left"><font face="Times New Roman" size="3">Check Booking Code</font><br/>
</p>
<form>
<p>&nbsp;&nbsp;</p>
</form>

<table>
<thead>
<tr>
<th>Name: addbooking.asp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Written by: Baistang Liu</td>
</tr>
<tr>
<td>Purpose: Insert booking information into booking table</td>
</tr>
</tbody>
</table>

```html
<DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!-- page created by Baistang Liu on Jun 19, 2001  -->
<HEAD><TITLE>Addbooking.asp</TITLE></HEAD>
<style>
@text-decoration:none;
</style>
<SCRIPT language="JavaScript">
function MM_swapImage(i,im,im2) { if (this.offsetWidth&gt;0) {
var L=imageArray[i].src; imageArray[i].src=imageArray[i].alt; imageArray[i].alt=L; } else {
var L=imageArray[i].alt; imageArray[i].alt=imageArray[i].src; imageArray[i].src=L; }
}
</SCRIPT>

var imageArray=document.MM_img; for(i=0;i&lt;imageArray.length;i++) imageArray[i].xsrc=imageArray[i].src;
```
```
CREATE CONNECTION = SERVER=des;driver=(Microsoft ODBC for Oracle);UID=1;PWD=1;

Dim objConnection

' Create the connection object
Set objConnection = Server.CreateObject("ADODB.Connection")

objConnection.ConnectionString = cConnection

objConnection.CursorLocation = 3 * adUseClient

objConnection.Open

' Return the new connection back to the calling routine
Set GetDBCConnection = objConnection

End Function

' Add booking information into the booking table
```
```c
Const cProcName = "call customer_pkg.AddBill( resultset 100, L_pktID, L_pktCode, L_custID, L_lttID))"

* Set up the command object as call the Stored Procedure

Set objCustomerCommand = Server.CreateObject("ADODB.Command")

objCustomerCommand.ActiveConnection = GetDBConnection()

objCustomerCommand.CommandText = cProcName

objCustomerCommand.CommandType = 1  ** asCmdText

* Execute the Stored Procedure.

Set objCustomerResultSet = objCustomerCommand.Execute()

* Close everything

objCustomerResultSet.Close

Set objCustomerResultSet = Nothing

Set objCustomerCommand = Nothing

GetCustomerListOptions = strResult

End Function

%>

<!--

</TABLE> <BR><BR>
<TABLE cellSpacing=0 cellPadding=0 width=540 border=0>
<TBODY>

</TD> <TD width=400><TD>
<TD width=500>
<P align="center"> <FONT face="Verdana, Arial, Helvetica, sans-serif" color="#000000 size=1">California Seller of Travel #1009907-40</FONT> </P> <P align="center"> <FONT face="Verdana, Arial, Helvetica, sans-serif" color="#000000 size=1"> </FONT> </P> <TABLE width=500>
<TBODY>

</TD>

</TD>

</TBODY>

</TABLE>

</TD>

</TD width=166>
</CENTER><A href="http://www.usstate.com"><IMG height=38
hipcote=10 src="CTS Home Page_files/usstate.jpg" width=80
border=0></A> <CENTER><TD>
<TD width=166>
</CENTER><A href="http://www.iesa.org"><IMG height=31
hipcote=10 src="CTS Home Page_files/IESA.jpg" width=70
border=0></A> <CENTER><TD>
<TD width=166>
</CENTER><A href="http://www.pass.org"><IMG height=37
hipcote=10 src="CTS Home Page_files/PASS.jpg" width=55
border=0></A> <CENTER><TD>
<TD>
<TD width=166>
</CENTER><A href="http://www.airchiva.com.cn"><IMG height=69
hipcote=10 src="CTS Home Page_files/AirChina.jpg" width=75
border=0></A> <CENTER><TD>
<TD width=167>
hipcote=10 src="CTS Home Page_files/C listen.jpg" width=100
border=0></A> <CENTER><TD>

74
```
*Name: addCustomer.html

*Written by: Balitang Liu

*Purpose: create customer registration form

```html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>New Customer Registration</title>
</head>
<body>

<h1>Customer Registration Form</h1>

<form action="addCustomer.php" method="post">
  <label for="firstName">First Name:</label><br>
  <input type="text" id="firstName" name="firstName"><br>
  <label for="lastName">Last Name:</label><br>
  <input type="text" id="lastName" name="lastName"><br>
  <label for="email">Email Address:</label><br>
  <input type="text" id="email" name="email"><br>
  <label for="phoneNumber">Phone Number:</label><br>
  <input type="text" id="phoneNumber" name="phoneNumber"><br>
  <label for="address">Address:</label><br>
  <input type="text" id="address" name="address"><br>
  <label for="city">City:</label><br>
  <input type="text" id="city" name="city"><br>
  <label for="state">State:</label><br>
  <input type="text" id="state" name="state"><br>
  <label for="zipCode">ZIP Code:</label><br>
  <input type="text" id="zipCode" name="zipCode"><br>
  <label for="country">Country:</label><br>
  <input type="text" id="country" name="country"><br>
  <input type="submit" value="Submit">
</form>

</body>
</html>
```
<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Name</td>
<td>text</td>
<td>Name of the person</td>
</tr>
<tr>
<td>Last Name</td>
<td>text</td>
<td>Family name of the person</td>
</tr>
<tr>
<td>DOB</td>
<td>text</td>
<td>Date of Birth</td>
</tr>
<tr>
<td>SSN</td>
<td>text</td>
<td>Social Security Number</td>
</tr>
<tr>
<td>Address</td>
<td>text</td>
<td>Physical Address</td>
</tr>
<tr>
<td>City</td>
<td>text</td>
<td>City of residence</td>
</tr>
<tr>
<td>State</td>
<td>text</td>
<td>State of residence</td>
</tr>
<tr>
<td>Zip</td>
<td>text</td>
<td>Zip Code</td>
</tr>
<tr>
<td>Phone</td>
<td>text</td>
<td>Contact Number</td>
</tr>
<tr>
<td>E-Mail</td>
<td>text</td>
<td>Email Address</td>
</tr>
</tbody>
</table>

<form>
<p align="left" style="margin-top: 2; margin-bottom: 2">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&n...
function MM_swapImage() { var p = new Array(); document.MM_sr = new Array(); for(i=0;i<(p.length-2);i++) p[i] = new Array(); return false; }

<SCRIPT>

</SCRIPT>

<LINK href="CTS Home_page_files/style.css" rel="stylesheet">

<STYLE type="text/css" id="slide2">

FONT-WEIGHT: bold; FONT-SIZE: 11px; COLOR: #000; LINE-HEIGHT: 14px; FONT-FAMILY: Verdana, Arial, Helvetica, sans-serif
</STYLE>

</BODY>

</HTML>
* Name: GetDBConnection

* Description:

* Returns an open ADO Connection object.

******************************************************************************

Private Function GetDBConnection

Const cConnection = "SERVER=desgn\driver\(Microsoft ODBC for Oracle); UID=lu; PWD=literal;"

Dim objConnection

* Create the connection object

Set objConnection = Server.CreateObject(“ADODB.Connection”)

objConnection.ConnectionString = cConnection

objConnection.CursorLocation = 1 * ”useClient

objConnection.Open

* Return the new connection back to the calling routine

Set GetDBConnection = objConnection

End Function

Private Function AddNewCustomer()

Const cProcName = "call customer_Pkg.AddCustomer() readset 100,
I_custID, I_FirstName, I_LastName, I_DOB, I_SSN, I_Addr, I_city, I_state,
I_zipCode, I_phone, I_email)"

* Set up the command object to call the Stored Procedure

Set objCustomerCommand = Server.CreateObject("ADODB.Command")

objCustomerCommand.ActiveConnection = GetDBConnection()

objCustomerCommand.CommandText = cProcName

objCustomerCommand.CommandType = 1 * ”adCmdText

* Execute the Stored Procedure.

Set objCustomerResultSet = objCustomerCommand.Execute()
Close everything

objCustomerQueryString.Close

Set objCustomerResultSet = Nothing

Set objCustomerCommand = Nothing

---

<TBODY>
  <TR>
    <TD width=166>
      <CENTER><A href="http://www.aerotac.com/" target="_blank"><IMG height=58
        src=CTS Home_page_files/aero.png" width=80
        border=0</A> <CENTER></TD>
    </TR>
  <TD width=167>
      <CENTER><A href="http://www.airusa.org/" target="_blank"><IMG height=51
        src=CTS Home_page_files/AirUSA.gif" width=70
        border=0</A> <CENTER></TD>
    </TR>
  <TD width=165>
      <CENTER><A href="http://www.pata.org/" target="_blank"><IMG height=57
        src=CTS Home_page_files/PATA.gif" width=55
        border=0</A> <CENTER></TD>
    </TR>
  <TD width=166>
      <CENTER><A href="http://www.airchina.com.cn/" target="_blank"><IMG height=69
        src=CTS Home_page_files/AirChina.gif" width=75
        border=0</A> <CENTER></TD>
    </TR>
  <TD width=167>
      <CENTER><A href="http://204.127.117.130/" target="_blank"><IMG height=65
        src=CTS Home_page_files/Cowtherm.jpg" width=100
        border=0</A> <CENTER></TD>
    </TR>
  <TD width=163>
      <CENTER><A href="http://www.cia.com/" target="_blank"><IMG height=14
        src=CTS Home_page_files/US.gif" width=150
        border=0</A> <CENTER></TD>
    </TR>
  </TABLE></TR></TABLE></TBODY></BR></CENTER>

<BR>Booking questions? <A href="mailto:CTSUSA@aol.com">email us</A>!
</P>

<BR>Questions for the <A href="mailto:mike@fsnation.com">Webmaster</A>.
</P>

---

**Name: Admin Login Page**

**************************************************************************
<DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

/* Admin Login Page designed by RayChung Lin on June 15, 2001 */
<table>
<thead>
<tr>
<th>Day</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Day</td>
<td>A Day tour of the most popular European destinations.</td>
</tr>
<tr>
<td>Two Days</td>
<td>Two Days trip to the best cities in Europe.</td>
</tr>
<tr>
<td>A Week</td>
<td>A Week-long journey exploring the continent.</td>
</tr>
</tbody>
</table>

Please select your choice:
- A Day
- Two Days
- A Week
- A Month

Select your option:
- Submit

Questions for the Webmaster?
"* Return the new connection back to the calling routine

Set GetDBConnection = objConnection

End Function

%>

Private Function BuildNewBookingTable

Const cProcName = "[call Customer_Pkg.SelectNewBooking\$, irequest 100, O_BkgID, O_BkgCode, O_custID, O_JndID)]"

******************************************************************************
* The position of parameters in the stored proc
******************************************************************************

Const cTimeRangeParam = 0
Dim objCustomerCommand

* Command object to execute Stored Procedure

Dim objCustomerResultSet

* ResultSet containing list of new booking

******************************************************************************
* Set up the command object
******************************************************************************

Set objCustomerCommand = Server.CreateObject("ADODB.Command")
objCustomerCommand.ActiveConnection = GetDBConnection()

objCustomerCommand.CommandText = cProcName
objCustomerCommand.CommandType = 1  ** adCmdText

******************************************************************************
* Set the direction and value
******************************************************************************

objCustomerCommand.Parameters(cTimeRangeParam).Direction = 1  ** adParamInput

objCustomerCommand.Parameters(cTimeRangeParam).Value = Request.Form("TimeRange")

* Set the input param
*Execute the stored proc

Set oCustomerResultSet = oCustomerCommand.ExecuteNonQuery

* Go thru the result set and build the resulting HTML table

Do While Not oCustomerResultSet.EOF

strResult = strResult & "<TR>
    & "<TD>" & oCustomerResultSet.Fields("o_BkgID") & "</TD>"
    & "<TD>" & oCustomerResultSet.Fields("o_BkgCode") & "</TD>"
    & "<TD>" & oCustomerResultSet.Fields("o_CustID") & "</TD>"
    & "<TD>" & oCustomerResultSet.Fields("o_firstID") & "</TD>"
    & "</TR>

oCustomerResultSet.MoveNext

Loop

* Close everything

oCustomerResultSet.Close
Set oCustomerResultSet = Nothing
Set oCustomerCommand = Nothing

* Return back the HTML

BuildNextBookingTable = strResult

End Function

%>
Please click on appropriate links to check the booking information!
<TD align="top" width="120">
<TABLE cellSpacing=0 cellPadding=1 width="116" border="0" valign="top">
</TD>
</TD>
</TR>

<TD width="50" &nbsp;>&lt;TD width="50"&gt;&lt;/TD&gt;
</TD>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TR>

<TD width="50" &nbsp;>&lt;TD width="50"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>

<TD align="top" width="116">
&lt;TD align="top"&gt;&lt;/TD&gt;
</TD>
</TD>
</TR>
Private Function GetDBConnection

Const cConnection = "SERVER=\server_name;\driver={Microsoft ODBC for Oracle};UID=\username;PWD=\password;"
Dim objConnection

'* Create the connection object
Set objConnection = Server.CreateObject("ADOConn.Connection")
objConnection.ConnectionString = cConnection
objConnection.CursorLocation = 3 * adUseClient
objConnection.Open

'* Return the new connection back to the calling routine
Set GetDBConnection = objConnection

End Function

%>

'* Call the CheckID procedure to get their name

******************************************************************************
Const cProcName = "[call Employee_Fkg.CheckID (?,?1,?2)]"  
******************************************************************************

'* The position of parameters in the stored proc

******************************************************************************
Const cEmailParam = 0
Dim objCustIDCommand

" Command object to execute Stored Procedure

*******************************************************************************

" Set up the command object
*******************************************************************************
Set objCustIDCommand = Server.CreateObject("ADODB.Command")
objCustID.ActiveConnection = GetDConnection()
objCustIDCommand.CommandType = eProcName
objCustIDCommand.CommandType = 1 "** adCmdText

*******************************************************************************

" Set the direction and value for the input
*******************************************************************************
objCustIDCommand.Parameters(cEmailParam).Direction = 1 "** adParamInput

objCustIDCommand.Parameters(cEmailParam).Value = Request.Form("Email")

" Set the input param to the selected ID

*******************************************************************************

" All the others are output so set them so
*******************************************************************************
objCustIDCommand.Parameters(cCustIDParam).Direction = 2 "** adParamOutput

*******************************************************************************

" Execute the stored proc
*******************************************************************************
Call objCustIDCommand.Execute()

*******************************************************************************

" Now we can show the values
*******************************************************************************
%>
B. Bibliography


Aaron Walsh and John Fronckowiak, 1998, Java Bible, published by IDG Worldwide, Inc.


Francis, Kauffman, Llibre, Sussman and Ullman, 2000, Beginning Active Server Pages 2.0, published by Wrox Press Ltd

Oracle Corporation, 1999, Oracle University SQL and PL/SQL, published by Oracle Corporation

Don Gosselin, 2000, JavaScript, published by Thomson Learning

www.oracle.com  Oracle Cooperation website

www.microsoft.com  Microsoft Cooperation website

www.java.sun.com  Sun Microsystem website

Related courses

INFS 684 Database Design and Performance Analysis
Course link: http://www.courses.dsu.edu/Infs684/default.htm
Instructor: Dr. Ronghua Shan

INFS740-Economic Commerce
course link: http://ultra5-07nt.dsu.edu/Home/Courses/INFS740/
Instructor: Dr. Kuanchin Chen

INFS787 Information Retrieval
Course link: http://www.courses.dsu.edu/Infs787/default.htm
Instructor: Dr. Ronghua Shan

INFS755 Project & Change Management
Course link: http://www.homepages.dsu.edu/dennist
Instructor: Dr. Terry Dennis

INFS 750 Telecommunications Technology & Management
Course link: http://www.courses.dsu.edu/infs750
Instructor: Professor Minhua Wang

INFS788-Emerging Database Technologies
Course link: http://www.courses.dsu.edu/Infs788/default.htm

INFS741-Web Server Management
Course link: http://ultra5-07nt.dsu.edu/Home/Courses/INFS741/
Instructor: Dr. Kuanchin Chen