NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

ESTATE AGENT WEB SITE

Graduation Project
Com – 400

Submitted by: Selcuk DURAN (980687)

Submitted to: Assoc. Prof. Dr Rahib ABİYEV

Nicosia - 2004

ACKNOWLEDGEMENT

"I am very grateful to my parents and all of my teachers who gave me chance to finish The Near East University

I would like to thank my supervisor Ass Prof Dr RAHIB ABIYEV for his invaluable advice and belief in my work and myself over the course of this Graduation Project

I thank a lot to my family for their patience

And Thanks very much to all of my friends"

TABLE OF CONTENTS

CHAPTER 1

INTRODUCING ACTIVE SERVER PAGES

1.1 Introduction

Active Server Pages is an open, compile-free application environment in which you can combine HTML, scripts, and reusable ActiveX server components to create dynamic and powerful Web-based business solutions. Active Server Pages enables server side scripting for IIS with native support for both VBScript and JScript.

Or we can say, Active Server Pages (ASPs) are Web pages that contain *server-side scripts* in addition to the usual mixture of text and HTML tags. Server-side scripts are special commands you put in Web pages that are processed before the pages are sent from the server to the web-browser of someone who's visiting your website. When you type a URL in the Address box or click a link on a webpage, you're asking a web-server on a computer somewhere to send a file to the web-browser (also called a "client") on your computer. If that file is a normal HTML file, it looks the same when your web-browser receives it as it did before the server sent it. After receiving the file, your web-browser displays its contents as a combination of text, images, and sounds. In the case of an Active Server Page, the process is similar, except there's an extra processing step that takes place just before the server sends the file.

1.2 Scripting

Programming Active Server Pages with VBScript, client-side scripting and server-side scripting have different missions in life. Client-side scripts most often add improved user interface and data validation (when HTML forms are used). Server-side scripts, especially in Active Server Pages, are practically unlimited; but they are primarily used to capture business rules and access to data.

The important thing to stress here, however, is that server-side can, if properly implemented, create client-side scripts. One of the most important questions in Internet development is the one that makes you choose between programming to the broadest audience and programming for the richest on-line experience. To the extent that you choose the former, server-side scripting is important for two reasons:

1. Server-side scripts can sense the capabilities of requesting client programs.
2. They can be as powerful as you, the designer, want, regardless of how thin the client is.

For example, the thinnest client on the Internet is the one that cannot render graphics of any kind. The ALT parameter of the IMG tag in HTML originally was intended to help such clients interpret important parts of the screen that they otherwise couldn't see, by describing the area in words instead of an image. With an Active Server Page, your application can sense when such a browser (for that's what these kinds of programs are-as opposed to Web client programs that have more processing power) is making a request of your Web site. You can then present such graphics-challenged browsers with whole paragraphs, not merely short expressions, to give hem as much information as possible, given their inherent limitations. In today's Internet, a major difference between Web clients brands is whether they recognize ActiveX controls or not. Again, the Active Server Page doesn't care one way or the other. If it senses the ability to interpret ActiveX controls, it presents them; otherwise, it includes static images (or text, if necessary). Of far greater importance than these mundane issues is the fact that Active Server Pages promote a new level of processing power into the Web server. It is critical to remember that the Web server was never designed to be an application server. It was designed to deliver HTML. This remains its primary mission, even on the Active Platform, but with Active Server Pages, this design constraint ceases to be a constraint at all.

The scripts that are contained in Active Server Pages, especially those driven by Active Server components, bring virtually all the power of traditional client/server programming to the Web server. Indeed, to the extent that Active Server components are utilized, Active Server Pages can do things that even the most sophisticated client/server programs can't. That's a pretty strong statement.

1.3 Components

Components may be the single most important feature of Active Server Pages. Their importance to ASP is understandable when you step back and see how pervasively Microsoft has embraced components in virtually everything they create. Everything from the Internet Explorer to Windows NT 5.0 has been "componentized" by Microsoft engineers. Components give programmers many advantages, including lower development time and cost, added flexibility, easier maintenance, and most important, easy scale ability. For the ASP development community,

on the server-side, server components are either intrinsic to the Active Server or they are user-defined. On the client-side, ActiveX controls provide functionality similar to server components.

1.4 Active Server Components

Active Server Components basically do two things. First, they directly expose operating system services to your Active Server Pages. Second, they encapsulate business rules in a way that is extremely easy to program. Perhaps even more important in the long run, Active Server Components are easy to create. That is, by using programming tools and environments optimized to work with the Active Platform, writing sophisticated server components is no longer the province of the advanced programmer.

There is a truism in programming that the best programmers are users. Active Server components will prove that not only to be true but important, as well. In the summer of 1996, it was estimated that the number of lines of Visual Basic code finally exceeded the number of lines of code written in COBOL, the perennial champ. Perhaps the biggest reason Visual Basic is so prolific is that users, not professional programmers, wrote these "extra" lines of code. Active Server component development will bring the same ease of programming to the Internet that Visual Basic brought to creating Windows programs. The Browser Capabilities component is the component that permits an Active Server Page to determine what kind of browser or Web client program is making a request. It makes this determination by looking to the User Agent HTTP header and looking up the identified browser in the browscap.ini file. All of the listed features of the browser are exposed as properties of the Browser Capabilities component.

There is one Active Server Component that may keep you up nights, though. It's the Database Access component, and it exploits an operating system service of earthshaking importance: objects in the directory system. Actually, the earth won't shake until Windows NT 5.0 ships in 1997; at that time, ActiveX Data Objects (ADO) will be incorporated into the Windows NT Directory Services. That is, the directory system will be able to be managed like a database. Files become database objects with properties that will be exposed to ADO. You can already see what this will look like when you select the Properties menu option of a file on your Windows Desktop. By the way, these directory services aren't restricted to the Windows Explorer and the local file system; they reach out to every file system on the Internet!

We mentioned that a key design goal of the ADO team was to enable universal access to information-they do mean *universal.* To ADO, it won't matter if the data is a record in an ODBC database or a message stored in Exchange Server. It won't matter if the data is stored on your own

hard drive or on one in the Smithsonian. ADO will find it and present it to your application (possession is no longer nine-tenths of the law). Again, this is the logical conclusion of the Web. The Web doesn't let you take possession of HTML. ADO doesn't let you possess the data, either; it just makes it available to your application.

Now, imagine programming when most of the work done by your applications is done with the aid of other peoples' server components. Whether you're using a server component to access an interactive feature in your Web site or you access network functionality in Windows NT 5.0, you will be able to do far more programming of the real task at hand. No more time wasted doing things that every other programmer in the world is doing at the same time you are. Even if the objects exposed by Active Server components don't qualify as "true" objects in the minds of the purists, the kind of object-centric programming that will become commonplace in Active Server Pages development will have an impact great enough that most of us will forget about polymorphism and inheritance..

1.5 ActiveX Controls

ActiveX controls are used like server components, only on the client side. That is, you instantiate an ActiveX control in a client-side script with the OBJECT tag, and then you manipulate this control through its exposed properties and methods. Most ActiveX controls enhance the user interface of your Web applications, but some can simply return a value directly to your application. For example, you can write an ActiveX control that makes a complex calculation from given inputs. The control would receive the inputs through its properties, and the resulting calculation would be returned to the calling application through a separate property.

On the other hand, Active Server components never have a user interface. They are designed to render services to your server application for the purpose of producing standard HTML output. In other words, Active Server Pages are never used directly by people. Active Server Pages produce the HTML that users see, and that HTML may include ActiveX controls. So sensing browser capabilities or manipulating text files or providing HTML source code with a randomly selected image or filling the controls on an HTML form with data from a database are all examples of the usefulness of server components.

You may be tempted to suggest that Microsoft also wants you to use ActiveX controls for self-serving reasons, but this allegation carries less weight now that the Open Group is responsible for the standard.

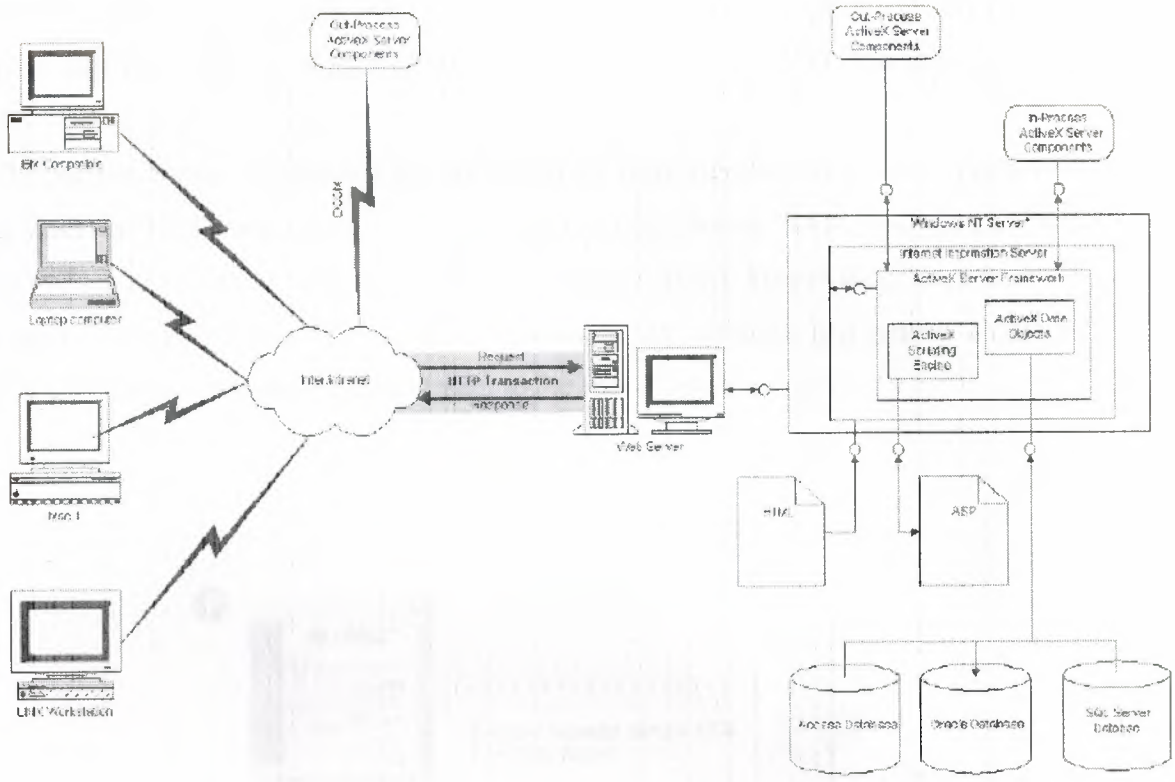

Figure 1.1 The programming environment of the Active Server is both rich and accessible to all programming skill levels.

1.6 Seeing Where ASP and HTTP Fit Together

There really are three entities involved in an HTTP transaction: the Web client, the Web server, and the human being. The Web client and Web server communicate using HTTP *headers*. These are collections of data that the client and server exchange with one another to ensure that, regardless of the contents of the body of the HTTP transaction, the entire transaction remains coherent and complete. The data displayed to the human being is transmitted from the Web server to the Web client, and the Web client transfers the text and the interpreted HTML source code to the screen or printer, so the human can read it.

Active Server Pages permit the developer to affect all facets of the HTTP transaction. The ASP objects known as Request and Response interact with the HTTP body and headers, respectively. This feature gives the ASP developer almost unlimited flexibility in management of interaction on the Web. For example, using these two objects lets the developer authenticate secure HTTP transactions and control the contents of the STATUS header, blocking access to

requested content when such access would violate established security policy. Even complex authentication schemes can be implemented using new headers defined just for your ASP application.

The Active Server is implemented as an ISAPI filter running under IIS. Whenever a Web client makes an HTTP request of a Web server, the Active Server ISAPI filter gets a chance to intercept the request. If the request is for an .asp file, the Active Server takes over from IIS, parses the entire file from top to bottom, processes the server script(s), and returns an HTML output file to IIS. IIS then returns this data stream to the requesting Web client.

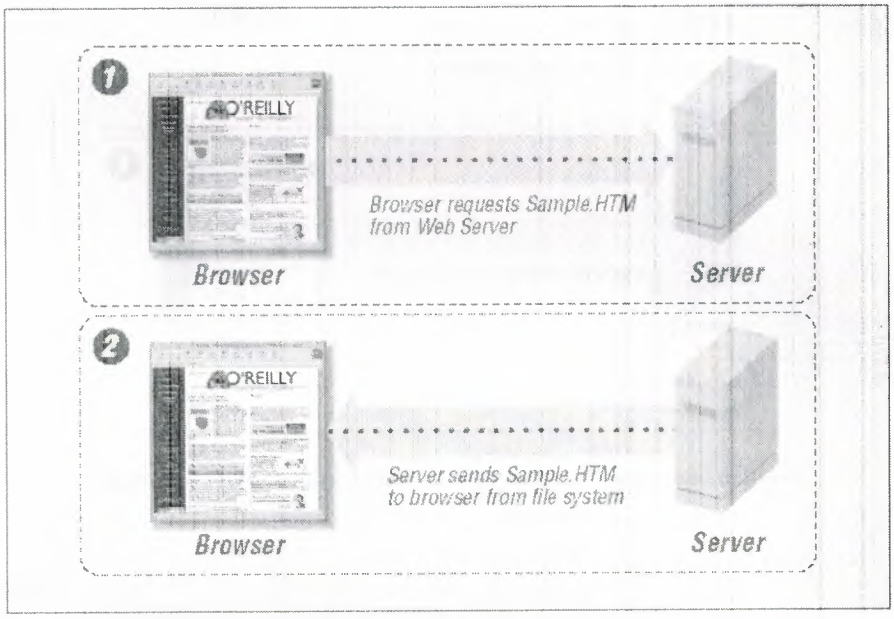

Figure 1.2 Static web content: request and delivery

Figure 1.3 Dynamically generated web content: request and delivery.

1.7 Special Cases

With sufficient experience, you may find that there's nothing beyond your reach with ASP extending your grasp. This new power won't come without exacting a cost, however. To really improve your reach with Active Server Pages you will have to meet the following two challenges, at least.

1. .asp files can populate client-side scripting objects with data that is accessed through ADO.
2. They can be used to generate data inside the Microsoft Layout Control's .alx file.

1.7.1 Dynamic Client-Side Scripts

The first challenge presents itself when the server is called to create a dynamic client-side script. The most frequent occurrence of this almost certainly will be in filling out on-line forms. For example, say you have an HTML FORM with SELECT tags and TEXT fields in it. Further

suppose that the specific variables displayed in these controls are stored in your database. The OnLoad event of your scripted page would normally populate the SELECT tag. With ASP, the server-side script would first fetch the SELECT options from the database, and it would then be able to write the client-side script that would run when the OnLoad event fired. The result is a dynamic SELECT tag.

1.7.2 HTML Layout Controls

Once you get past the more common dynamic HTML challenge, you will likely be confronted by the second challenge: using the ActiveX Layout Control in your .asp file. The trick is to give the file created in the ActiveX Control Pad an .asp extension, instead of the standard ALX value. There are other requirements that have to do with protecting the .asp delimiters embedded in the .alx/.asp file, but details dictate a prerequisite knowledge of .asp syntax that you don't learn until Chapter 11, "Building a Foundation of Interactivity with *Request* and *Response* Objects."

1.8 Understanding the Structure of Active Server Pages

There is no structure, per se, in an .asp file that isn't already there in the structure of the HTML, Visual Basic, or JavaScript code. In this respect, .asp files are not really programs. Indeed, a single .asp file can implement any combination of supported scripting engines, using languages as diverse as Perl and Rexx to Visual Basic and JavaScript. ASP is an "ecumenical" programming environment.

1.8.1 HTML, All by Itself

It is acceptable, though not necessarily recommended, to rename you HTML files with the .asp extension and turn them all into Active Server Pages. That's all that's required to make an ASP application. If you only want to control more of the HTTP headers in your HTML files, then you may see minimal .asp source code in those renamed HTML files. But if you want to turbo-charge those sluggish old HTML files, or if you want to stop maintaining two versions of your Web site (one for the interactive-impaired), then read on.

1.8.2 HTML Mixed with .asp Source Code

Once you choose to add .asp source code to your HTML files, you have to make several more choices. If you are silent, the Active Server Engine makes a few of these choices on your

behalf. The choices fall into two categories: to use scripting or not and, if so, what kind(s) of scripting.

For the purposes of this discussion, .asp source code consists of either *native ASP commands* or *scripting commands*. Native commands are those that access Active Server Engine objects and components. Scripting commands rely on a particular syntax, as well. This means that you have to tell the Active Server Engine which language to use to interpret the commands. If you are silent, the engine will use VBScript by default.

This choice is not trivial when you are using Active Server Pages to write client-side scripts. As soon as you opt for this feature in your Web site, you're back to square one: are you writing to a captive audience such as an intranet, where all the client programs are the same brand and version? Even if all the browsers are the same brand and version, do they all support VBScript, or will you have to rely on the more ubiquitous JavaScript?

As noted in the introduction to this section, you don't have to choose one scripting engine. Choose the ones that suit your needs. If you have a nifty Perl program that you'd like to use, use it. If most of your server-side scripting will be done in VBScript because that's the language in which you are most fluent, use it. And if you need a generic, client-side scripting engine, use JavaScript, while you're at it.

1.8.3 Server-Side Includes

Server-Side Includes is powerful tools for programmer productivity. In a sense, they are the most basic kind of reusable code. Their primary purpose is to insert text file contents into .asp files. Server-Side Includes can contain other Server-Side Includes, so you can stuff an incredible amount of text into an .asp file with a single command.

Because Server-Side Includes are included in your .asp files before any of the files' ASP commands are executed, Server-Side Includes can't do anything fancy, such as looking up database records. They can, however, call other Server-Side Includes. Server-Side Includes inserts text in exactly the same place in your file as they are located. In other words, they replace themselves at runtime. This distinction can be important when the resulting text has a particular role to play and that role has a particular place in the file to play it. At other times, this is not so important. One of the most common uses for the Server-Side Include is when you need to refer to constants in your .asp source code.

A final point about Server-Side Includes is that they really don't add any marginal overhead. In a UNIX shop, however, .html files are usually not opened before they are sent on to the client program. But to process a Server-Side Include the server must open the .html file and the Server-Side Include file. It must then insert the text in the Server-Side Include into the .html file at the proper location finally; it must close the .html file and send it on to the Web client. Under the Active Server, the .asp file has to be opened anyway, so the extra effort of inserting the text is negligible. Anyway, this entire file I/O is processing in the address space of Windows NT, so even in the worst case, the overhead of processing .asp files in this way is nothing compared to the power you get in the bargain.

CHAPTER 2

INTEGRATING VBSCRIPT INTO HTML

2.1 A Brief History of Microsoft's BASIC Languages

The history of the BASIC language is a good place to start when putting VBScript and Active Server Pages development into perspective. The Beginners All Purpose Symbolic Instruction Code, more commonly known as BASIC, was developed in 1964 at Dartmouth College by Kenney and Kurtz. It was initially designed to provide students with an easy to understand procedural language, which would be a stepping stone to more powerful languages like FORTRAN. In the intervening 30+ years, a great deal has happened to this introductory computer language.

The language has grown and become more feature-rich over the years due mainly to its vast acceptance in the marketplace. To understand the evolution of the BASIC language and how it has become the default language of Active Server Pages scripting, we begin our story in 1975 when a young man named Bill Gates, was attending Harvard. Attracted by an article about the forthcoming M.I.T.S. Altair computer, Paul Allen and Bill Gates developed a version of BASIC that would run on the Altair and was eventually licensed to M.I.T.S. for their Altair computer. When version 2.0 was released later that same year, it was available in two versions, a 4K and an 8K. Imagine the entire development system implemented in 4096 bytes! Today, you would be hard-pressed to find a Microsoft Word template that is that small. Basic was the first product ever sold by Microsoft. Two years later, after porting their version of BASIC to other platforms (CP/M, for example) the exclusive license with M.I.T.S. for Microsoft Basic ended. In 1979, Microsoft released MS-Basic for the 8086, a 16-bit product.

Bill Gates won the opportunity to provide the operating system for the new IBM personal computer after IBM's courting of Digital Research Inc. to license their CP/M operating system failed. Microsoft licensed the SCP-DOS operating system and modified it to run on the IBM-PC. The MS-DOS operating system version 1.0, bundled with MS-BASIC was the engine driving the beginning of the personal computer revolution.

Over the years, Microsoft saw how attractive BASIC was and created a compiler for the language in the form of QuickBasic. QuickBasic reigned supreme until version 4.5, when it was replaced with PDS Basic (Professional Development System).

We had no idea in that spring of 1991 that many of our lives were going to change so dramatically. Visual Basic was announced at the Windows World '91 conference on May 20, 1991. The Visual Basic environment was to provide graphical application development and an integrated debugger and to create compiled executable windows programs, all using the BASIC language. Many Windows developers still remember the first time that they used Visual Basic version 1.0. After thrashing their way though learning the ins and outs of the C language and building Windows applications with Microsoft C and the SDK, they couldn't believe the power inherent in this innocuous little visual development package.

Visual Basic for Windows was followed by Visual Basic for DOS. When the DOS version came out, there were (and still are) many programmers with DOS machines in our companies. The DOS version of VB addressed the RAD methodology on the DOS platform. Even though the product never made it past version 1.0, it was a useful tool for creating graphical applications for the DOS environment.

By the time Visual Basic version 4.0 was released in 1995, countless numbers of programmers were hooked on the Visual Basic development environment. It's easy learning curve, intuitive interface, and bundled components, combined with incredible extensibility and its tightly integrated environment make it the logical choice for millions of developers each day.

2.2 Visual Basic Scripting Edition

The scripting edition of Visual Basic is a smaller subset of the Visual Basic for Applications language. It is intended for use in Inter/intranet application development and is currently supported in Microsoft Internet Explorer version 3.0 and above. It brings much of the power and flexibility of the Visual Basic language to the Internet and intranet. On the client side, there is the opportunity to interact with ActiveX controls to provide active and interesting content. On the server-side, the scripting language is used and integrated within HTML to provide a new level of functionality and ease of use in Web site development.

For VB or VBA programmers, the transition to Active Server Pages development using VBScript from a traditional client/server environment will be less a challenge of learning the idiosyncrasies of a sister language than a challenge of changing to the new net development paradigm. Programming in any language consists of expressions, statements, and procedures. The trick is to figure out how the language integrates with the environment in which it will be implemented. In the case of VB or VBA, the environment is the Windows operating system.

VBScript, on the other hand, will be implemented on the client, using ActiveX controls, as well as on the server in ASP, integrating a variety of components to create dynamic pages. You will be dealing with, not only the scripting language, but also its integration into HTML code. At first, having your code in pieces throughout the HTML page will take some getting used to. But, just as it was a struggle to master the VB IDE, you will master VBScript and Active Server Pages development.

If you are coming to Active Server Pages development from a strictly HTML background, you also will have a learning curve to climb. If you have been developing Perl or REXX scripts, the language features of VBScript will not be that foreign to you. Also, you have been used to adding additional tags as the HTML standard emerges. You can treat VBScript and the associated implementation as just some additional tags to integrate. But, be sure to utilize the new components that ship with Active Server Pages. This powerful set of ASP components includes such features as session and application management, and database connectivity. It would be very easy to use VBScript for some minor chores and revert back to the old CGI way of doing things for database access and other local processing tasks.

2.3 Client-Side Scripting

Client-side scripting refers to the scripts that are interpreted and executed in the client browser. When you are scripting for the client, you have access to the object model available within the browser.

There are a number of tools available to create client-side pages and their associated scripting. The ActiveX Control Pad is a good example of such tools. This Microsoft developed freely available product enables you to design Web pages, adding ActiveX controls and standard HTML fields at design time. The program then generates the HTML code to create the page. After the page has been created, you can edit the file and add scripting to provide such client-side features as field validation, custom responses to user actions, and a host of other capabilities inherent in the client's browser.

As mentioned previously, the opportunity for field validation of data at the client is an important feature of client-side scripting. You can have the page validate the data *before* it is sent to the server. This ensures that you will not receive a message immediately back from the server requesting you to provide complete, or correct, information. In addition to providing validation errors more quickly to the user, this also can reduce network traffic.

2.4 Server-Side Scripting

Now we get to the meat and potatoes of our scripting discussion. Server-side scripting occurs when the scripts within the page are executed at the server, before the page is ever sent to the client browser, as shown in Figure 2.1. This is an incredibly important distinction. It means that the server is responsible for generating the HTML that is ultimately sent to the client. You do not have to worry about the client connecting to a database, reading from a file, querying an on-line service, or any of the thousands of other actions that take place on the server to fulfill the client request.

Active Server Pages provides server-side scripting for the Internet Information Server Web server. In addition to enabling custom scripting to be developed, you can also integrate almost any ActiveX component (that doesn't require a user interface, of course) into your server scripts. This opens the door wide and enables a level of functionality that was difficult, if not impossible, to achieve with traditional methods of server-side processing.

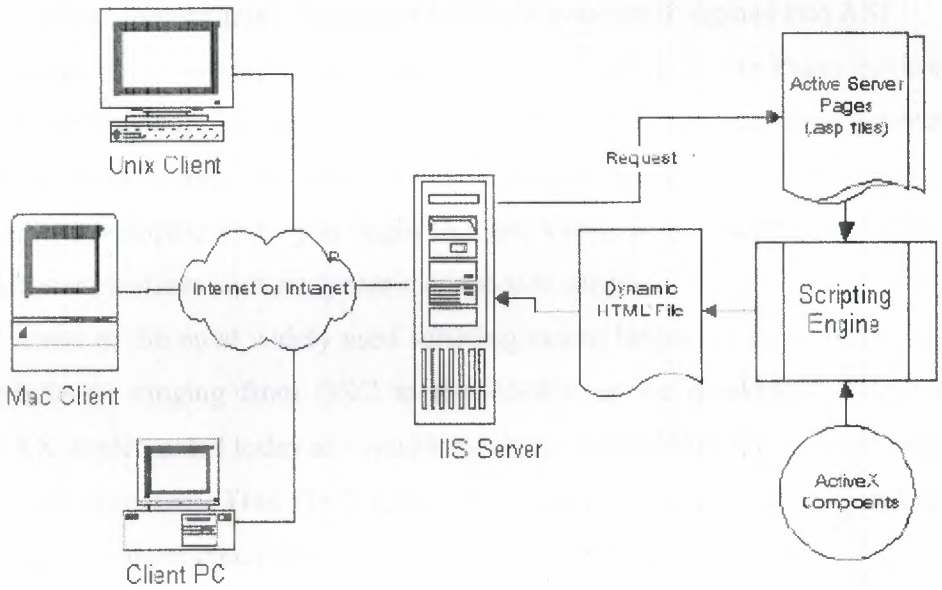

Figure 2.1 *Active Server Pages scripts execute on the server before passing the page to the client.*

Server-side scripting blocks are executed at the server when the ASP interpreter finds the scripting tag with the RUNAT attribute set to SERVER. In code, it looks like this:

<LANGUAGE=VBScript RUNAT=SERVER>

When an Active Server Pages page is requested, the server will call ASP. The .asp file is read through from top to bottom. Any scripting that needs to execute on the server is performed, and then the dynamically created HTML page is sent back to the client.

Notice that the code in the CheckField subroutine creates a message box to respond to user input. If you were to mistakenly add the RUNAT=SERVER to the client code, the message box would never be shown on the client because at the server, there is no interface to show a message box upon. But, what you could do at the server is generate custom messages based upon the time of day, or based upon data in a database and pop those messages in the client browser when the validation takes place. You can do this by dynamically generating client-side scripts from the server.

2.5 Java Script, REXX, and Other Scripting Languages

As discussed in the section "Changing the Primary Scripting Language," the default language of Active Server Pages is VBScript. Java Script is also supported by ASP "out-of-the-box." The OLE object model for scripting engines, which Active Server Pages supports, enables you to easily integrate other scripting languages and their associated engines into ASP.

The ability to host a variety of languages within the Active Server Pages environment is an incredibly powerful feature. If you are a developer with years of experience generating Perl scripts, there is no need to forgo all of that valuable knowledge. You can immediately become productive in ASP development. As you begin to learn VBScript or JScript you will be able to incorporate additional features such as dynamic client-side scripting.

REXX is one of the most widely used scripting/macro languages in existence today. It is available on platforms ranging from OS/2 to the AS/400 to the mainframe. There are even versions of REXX implemented today as visual languages. VisPro/REXX is one such example of a visual REXX environment. This OS/2 application provides an easy to use and incredibly powerful visual development metaphor, leveraging the REXX language.

For the countless REXX developers working in development today, the ability to plug a REXX scripting engine into Active Server Pages once again opens the gates wide to let the greatest number of people maximize their Inter/intranet development.

CHAPTER 3
INTRODUCTION TO SQL

3.1 A Brief History of SQL

The history of SQL begins in an IBM laboratory in San Jose, California, where SQL was developed in the late 1970s. The initials stand for Structured Query Language, and the language itself is often referred to as "sequel." It was originally developed for IBM's DB2 product (a relational database management system, or RDBMS, that can still be bought today for various platforms and environments). In fact, SQL makes an RDBMS possible. SQL is a nonprocedural language, in contrast to the procedural or third generation languages (3GLs) such as COBOL and C that had been created up to that time. The characteristic that differentiates a DBMS from an RDBMS is that the RDBMS provides a set-oriented database language. For most RDBMSs, this set-oriented database language is SQL. Set oriented means that SQL processes sets of data in groups.

Two standards organizations, the American National Standards Institute (ANSI) and the International Standards Organization (ISO), currently promote SQL standards to industry. The ANSI-92 standard is the standard for the SQL used throughout this book. Although these standard-making bodies prepare standards for database system designers to follow, all database products differ from the ANSI standard to some degree. In addition, most systems provide some proprietary extensions to SQL that extend the language into a true procedural language. We have used various RDBMSs to prepare the examples in this book to give you an idea of what to expect from the common database systems. (We discuss procedural SQL--known as PL/SQL--on Day 18, "PL/SQL: An Introduction," and Transact-SQL on Day 19, "Transact-SQL: An Introduction.")

3.2 A Brief History of Databases

A little background on the evolution of databases and database theory will help you understand the workings of SQL. Database systems store information in every conceivable business environment. From large tracking databases such as airline reservation systems to a child's baseball card collection, database systems store and distribute the data that we depend on. Until the last few years, large database systems could be run only on large mainframe computers. These machines have traditionally been expensive to design, purchase, and maintain. However,

today's generation of powerful, inexpensive workstation computers enables programmers to design software that maintains and distributes data quickly and inexpensively.

3.2.1 Designing the Database Structure

The most important decision for a database designer, after the hardware platform and the RDBMS have been chosen, is the structure of the tables. Decisions made at this stage of the design can affect performance and programming later during the development process. The process of separating data into distinct, unique sets is called *normalization*.

3.2.2 Today's Database Landscape

Computing technology has made a permanent change in the ways businesses work around the world. Information that was at one time stored in warehouses full of filing cabinets can now be accessed instantaneously at the click of a mouse button. Orders placed by customers in foreign countries can now be instantly processed on the floor of a manufacturing facility. Although 20 years ago much of this information had been transported onto corporate mainframe databases, offices still operated in a batch processing environment. If a query needed to be performed, someone notified the management information systems (MIS) department; the requested data was delivered as soon as possible (though often not soon enough).

In addition to the development of the relational database model, two technologies led to the rapid growth of what are now called client/server database systems. The first important technology was the personal computer. Inexpensive, easy-to-use applications such as Lotus 1-2-3 and Word Perfect enabled employees (and home computer users) to create documents and manage data quickly and accurately. Users became accustomed to continually upgrading systems because the rate of change was so rapid, even as the price of the more advanced systems continued to fall.

The second important technology was the local area network (LAN) and its integration into offices across the world. Although users were accustomed to terminal connections to a corporate mainframe, now word processing files could be stored locally within an office and accessed from any computer attached to the network. After the Apple Macintosh introduced a friendly graphical user interface, computers were not only inexpensive and powerful but also easy to use. In addition, they could be accessed from remote sites, and large amounts of data could be off-loaded to departmental data servers. During this time of rapid change and

advancement, a new type of system appeared. Called *client/server development* because processing is split between client computers and a database server, this new breed of application was a radical change from main frame based application programming. Among the many advantages of this type of architecture are

- Reduced maintenance costs
- Reduced network load (processing occurs on database server or client computer)
- Multiple operating systems that can interoperate as long as they share a common network protocol
- Improved data integrity owing to centralized data location

In *Implementing Client/Server Computing,* Bernard H. Boar defines client/server computing as follows:

Client/server computing is a processing model in which a single application is partitioned between multiple processors (front-end and back-end) and the processors cooperate (transparent to the end user) to complete the processing as a single unified task. Implementing Client/Server Computing A client/server bond product ties the processors together to provide a single system image (illusion). Shareable resources are positioned as requestor clients that access authorized services. The architecture is endlessly recursive; in turn, servers can become clients and request services of other servers on the network, and so on and so on.

This type of application development requires an entirely new set of programming skills. User interface programming is now written for graphical user interfaces, whether it be MS Windows, IBM OS/2, Apple Macintosh, or the UNIX X-Window system. Using SQL and a network connection, the application can interface to a database residing on a remote server. The increased power of personal computer hardware enables critical database information to be stored on a relatively inexpensive standalone server. In addition, this server can be replaced later with little or no change to the client applications.

3.2.3A Cross-Product Language

You can apply the basic concepts introduced in this book in many environments--for example, Microsoft Access running on a single-user Windows application or SQL Server running with 100 user connections. One of SQL's greatest benefits is that it is truly a cross-platform language and a cross-product language. Because it is also what programmers refer to as a high-

level or fourth-generation language (4GL), a large amount of work can be donehigher-level language 4GL (fourth-generation) language fourth-generation (4GL) language in fewer lines of code.

3.2.4 Early Implementations

Oracle Corporation released the first commercial RDBMS that used SQL. Although the original versions were developed for VAX/VMS systems, Oracle was one of the first vendors to release a DOS version of its RDBMS. (Oracle is now available on more than 70 platforms.) In the mid-1980s Sybase released its RDBMS, SQL Server. With client libraries for database access, support for stored procedures and interoperability with various networks, SQL Server became a successful product, particularly in client/server environments. One of the strongest points for both of these SQL Server powerful database systems is their scalability across platforms. C language code (combined with SQL) written for Oracle on a PC is virtually identical to its counterpart written for an Oracle database running on a VAX system.

3.2.5 SQL and Client/Server Application Development

The common thread that runs throughout client/server application development is the use client/server computing of SQL and relational databases. Also, using this database technology in a single-user business application positions the application for future growth.

3.3 An Overview of SQL

SQL is the de facto standard language used to manipulate and retrieve data from these relational databases. SQL enables a programmer or database administrator to do the following:

- Modify a database's structure
- Change system security settings
- Add user permissions on databases or tables
- Query a database for information
- Update the contents of a database

The most commonly used statement in SQL is the SELECT statement (see Day 2, "Introduction to the Query: The SELECT Statement"), which retrieves data from the database and returns the data to the user. The EMPLOYEE table example illustrates a typical example of a SELECT statement situation. In addition to the SELECT statement, SQL provides statements for

creating new databases, tables, fields, and indexes, as well as statements for inserting and deleting records. ANSI SQL also recommends a core group of data manipulation functions. As you will find out, many database systems also have tools for ensuring data integrity and enforcing security (see Day 11, "Controlling Transactions") that enable programmers to stop the execution of a group of commands if a certain condition occurs.

3.3.1 Popular SQL Implementations

This section introduces some of the more popular implementations of SQL, each of which has its own strengths and weaknesses. Where some implementations of SQL have been developed for PC use and easy user interactivity, others have been developed to accommodate very large databases (VLDB).

3.3.1.1 Microsoft Access

We use Microsoft Access, a PC-based DBMS, to illustrate some of the examples in this text. Access is very easy to use. You can use GUI tools or manually enter your SQL statements.

3.3.1.2 Personal Oracle 7

We use Personal Oracle7, which represents the larger corporate database world, to demonstrate command-line SQL and database management techniques. (These techniques are important because the days of the standalone machine are drawing to an end, as are the days when knowing one database or one operating system was enough.) In *commandline RÊl,* simple stand + [cedilla] one SQL statements are entered into Oracle's SQL*Plus tool. This tool then returns data to the screen for the user to see, or it performs the appropriate action on the database. Most examples are directed toward the beginning programmer or first-time user of SQL. We begin with the simplest of SQL statements and advance to the topics of transaction management and stored procedure programming. The Oracle RDBMS is distributed with a full complement of development tools. It includes a C++ and Visual Basic language library (Oracle Objects for OLE) that can link an application to a Personal Oracle database. It also comes with graphical tools for database, user, and object administration, as well as the SQL*Loader utility, which is used to import and export data to and from Oracle.

We chose the Personal Oracle7 RDBMS for several reasons:

- It includes nearly all the tools needed to demonstrate everything about the database.
- It is available on virtually every platform in use today and is one of the most popular RDBMS products worldwide.

3.3.1.3 Microsoft Query

Microsoft Query is a useful query tool that comes packaged with Microsoft's Windows development tools, Visual C++, and Visual Basic. It uses the ODBC standard to communicate with underlying databases. Microsoft Query passes SQL statements to a driver, which processes the statements before passing them to a database system.

3.3.1.4 Open Database Connectivity (ODBC)

ODBC is a functional library designed to provide a common Application Programming Interface (API) to underlying database systems. It communicates with the database through a library driver, just as Windows communicates with a printer via a printer driver. Depending on the database being used, a networking driver may be required to connect to a remote database.

The unique feature of ODBC (as compared to the Oracle or Sybase libraries) is that none of its functions are database-vendor specific. For instance, you can use the same code to perform queries against a Microsoft Access table or an Informix database with little or no modification. Once again, it should be noted that most vendors add some proprietary extensions to the SQL standard, such as Microsoft's and Sybase's Transact-SQL and Oracle's PL/SQL.

You should always consult the documentation before beginning to work with a new data source. ODBC has developed into a standard adopted into many products, including Visual Basic, Visual C++, FoxPro, Borland Delphi, and PowerBuilder. As always, application developers need to weigh the benefit of using the emerging ODBC standard, which enables you to design code without regard for a specific database, versus the speed gained by using a database specific function library. In other words, using ODBC will be more portable but slower than using the Oracle7 or Sybase libraries.

3.3.2 SQL in Application Programming

SQL was originally made an ANSI standard in 1986. The ANSI 1989 standard (often called SQL-89) defines three types of interfacing to SQL within an application program:

- Module Language: Uses procedures within programs. These procedures can be called by the application program and can return values to the program via parameter passing.
- Embedded SQL: Uses SQL statements embedded with actual program code. This method often requires the use of a precompiler to process the SQL statements. The standard defines statements for Pascal, FORTRAN, COBOL, and PL/1.
- Direct Invocation--Left up to the implementor.

Before the concept of dynamic SQL evolved, embedded SQL was the most popular way to use SQL within a program. Embedded SQL, which is still used, uses *static* SQL--meaning that the SQL statement is compiled into the application and cannot be changed at runtime. The principle is much the same as a compiler versus an interpreter. The performance for this type of SQL is good; however, it is not flexible--and cannot always meet the needs of today's changing business environments. Dynamic SQL is discussed shortly.

The ANSI 1992 standard (SQL-92) extended the language and became an international standard. It defines three levels of SQL compliance: entry, intermediate, and full. The new features introduced include the following:

- Connections to databases
- Scrollable cursors
- Dynamic SQL
- Outer joins

This book covers not only all these extensions but also some proprietary extensions used by RDBMS vendors. Dynamic SQL allows you to prepare the SQL statement at runtime. Although the performance for this type of SQL is not as good as that of embedded SQL, it provides the application developer (and user) with a great degree of flexibility. A call-level interface, such as ODBC or Sybase's DB-Library, is an example of dynamic SQL.

Call-level interfaces should not be a new concept to application programmers. When using ODBC, for instance, you simply fill a variable with your SQL statement and call the function to send the SQL statement to the database. Errors or results can be returned to the program through the use of other function calls designed for those purposes. Results are returned through a process known as the binding of variables.

CHAPTER 4

INTRODUCING ACTIVEX DATA OBJECTS

4.1 Introduction

Computing originally started on large machines with small terminals. With the advent of the personal computer and powerful workstations, much of the processing migrated to the desktop. The introduction of the Web server shifted the balance of power back to a centralized programming model. Today, Java and ActiveX controls have, yet again, enabled the desktop to reassert itself.

As important as this egalitarian trait of ASP is, everything done so far in the book is trivial when compared to the innovations in ActiveX that are covered in this part of the text. These innovations are called ActiveX Data Objects (ADO).

The apparent impediments of bandwidth and latency may actually be a blessing. Most of your ADO learning curve will probably occur on intranets-where bandwidth is measured in tens, if not hundreds, of megabits and where latency is practically zero. By the time we break the bandwidth bottleneck with Asynchronous Transfer Mode (ATM) switches, ASP/ADO development will be second nature to you. With those caveats in mind, get ready to take a deeper look into this radical new resource: ADO.

4.2 The Family Tree

Until Microsoft's acquisition of Fox Software in 1992, the database was conspicuously absent from Microsoft's desktop arsenal. Today, FoxPro (now matured into Visual FoxPro), Microsoft Access, and SQL Server round out a balanced database strategy. Each product is designed for a particular market. For example, Visual FoxPro uses an Indexed Sequential Access Method (ISAM) file format and is generally recognized as faster than Access but slower than SQL Server. As the other desktop application, Access uses a special file format that is accessed through the Jet Data Access Object (DAO) model. DAO gives the programmer direct access to the database structure, something not as easily done in Visual FoxPro. SQL Server, on the other hand, is a database server, not a desktop application. It is an industrial-strength application that is highly scaleable. All three products use ODBC to share data with each other.

There is one other thing that these three products have in common, setting them apart from ADO: Their ability to be deployed in a distributed network is defined by the Remote Data Object (RDO) specification found in Visual Basic 4.0's Enterprise Edition. Specifically, they

work in *connected* networks. The Internet, on the other hand, is a *connectionless* network-a packet-switched network.

In fact, Web servers are even more problematic for programmers, especially database programmers, because they are "stateless" as well as connectionless. They're called *stateless* servers because once they serve a client request, they forget that the client ever was served. The server does not keep track of anything going on with the client application.Web servers have short memories.

In the desktop world, when two applications communicate, they do so for a given period of time. For example, when an Access database is linked to external files, it tests for the presence of those files as soon as it is opened. As long as the Access database remains running, any interruption in the connection (for example, say that a Novell file server goes down) triggers an event in Access, and a warning flashes in the Access application that the connection has been broken. In SQL Server, the server is aware of the presence of a client application as long as that application is engaged in a transaction with the database server. The length of time of such a transaction is not necessarily as short as it is with a Web server.

A Web server knows about a client application only long enough to deliver a file, but an Active Server is different. While it, too, disconnects from the client as soon as a file is returned, it remains connected as long as necessary to do two things: first, process the database and produce a recordset; second, enable the application to make subsequent calls and still have access to previously returned data.

4.3 OLE DB

Before getting into the meat of ADO, take a closer look at the application program interface (API) on which ADO is built, OLE DB (OLE DB is Object Linking and Embedding applied to databases). ADO is the first Microsoft technology built on this exciting new initiative. If you thought that the Web spawned an avalanche of great software and business models on the Internet just wait until developers hear about OLE DB! And, as an ADO programmer, you're leading the way!

First, it's best to set the context with a quick review of the interface, the Open Database Connectivity (ODBC) specification. ODBC is a desktop SQL-based specification whose API foundation is built on the C language. This means that it was designed for relational database systems, the kind with which you all are accustomed to working. ODBC is a real workhorse, enabling developers to build systems that integrate Jet databases, ISAM files in old FoxBase+

databases, and even SQL Server tables into one coherent user interface. ODBC is the lingua franca of all these relational dialects meaning it is the linguistic bridge connecting the disparate native languages of each database management system.

OLE DB, on the other hand, is a specification based on a C++ API, so it is object-oriented. OLE DB consists of *data consumers* and *data providers*. Consumers take data from OLE DB interfaces; providers expose OLE DB interfaces.

ODBC now is a subset of OLE DB. Currently, Microsoft has developed an OLE DB data provider that enables access to the old relational data. In fact, under some circumstances, OLE DB can access ODBC data faster than DAO or RDO. This is because DAO and RDO have to pass through the ODBC layer, and OLE DB connects directly to relational data sources. Figure 4.1 shows subtle difference in the approach taken by ADO.

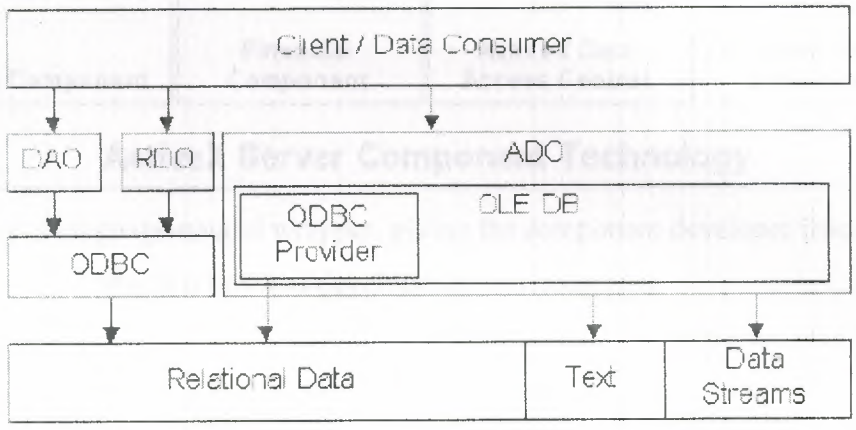

Figure 4.1 OLE DB simplifies the connection of your program to database information.

What's more, OLE DB can be used to extend the functionality of simple data providers. These more sophisticated and specialized objects are called *service providers*, and they can assemble everything needed by data consumers into a single table, regardless of data type (for example, ODBC, spreadsheets, e-mail messages, word processing documents, or file systems) or storage location (LAN, WAN, Internet, intranet). Service providers, therefore, are both consumers and providers. That is, a service provider could consume OLE DB interfaces and build a single table from that input; it then could expose OLE DB interfaces to a client application (such as an .asp file using ADO) for constructing its HTML output.

But OLE DB is, after all, a low-level specification. If you don't have time to learn a language lower down the food chain than Visual Basic, ActiveX is the answer. Specifically, ActiveX Data Objects use a language-neutral component technology to provide a high-level wrapper around the OLE DB API, which enables you to exploit all the power of OLE DB without

resorting to low-level programming. On the one hand, this means that all Microsoft programming languages can use ADO to access data. On the other hand, ActiveX components (ADO is but one example) can themselves be built using any language that complies with the Component Object Model.

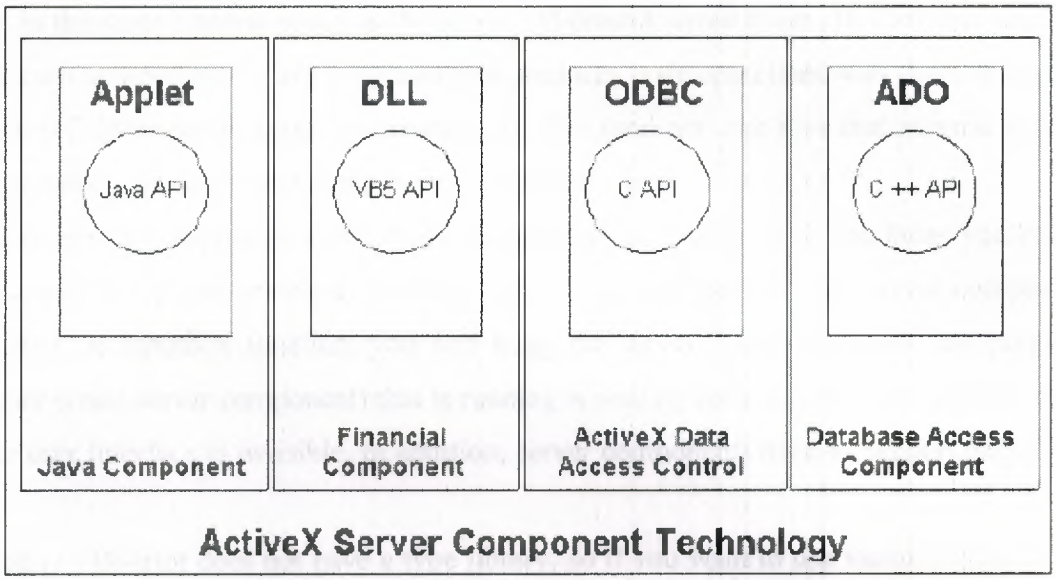

Fig 4.2 ActiveX is a language-neutral wrapper, giving the component developer free choice of the best breed of development environments.

To recap: ADO works just like DAO and RDO, only more efficiently, especially in a stateless and connectionless environment like the Internet/intranet. ADO, an ActiveX component technology, is based on a C++ API called OLE DB (which betrays its family heritage from ODBC, written in C). Because ADO is a component technology, it is highly extensible, tying ADO to Microsoft's programming paradigm for the next millennium.

4.4 Objects versus Components

The Active Server is a component of the Internet Information Server. Specifically, it is an Internet Server Application Programming Interface (ISAPI) filter. This means that it is a Dynamic Link Library file that becomes part of IIS as soon as the operating system starts it running. This modular design of everything Microsoft publishes these days is a model for the way an ASP developer should write applications.By exposing key ASP resources to the scripting

engine running on the server (e.g., VBScript, JavaScript, Perl, and others), the Active Server enables the ASP developer to extend the functionality of the Active Server itself.

Components have a similar function; viz., to extend the functionality of the Active Server, but they have one fundamental difference: they are .dll files that run separately from the Active Server, but in the same address space as the server. Microsoft wrote these .dll files, but you can create your own components in any language that produces code compliant with the Component Object Model (COM) specification. By creating .dll files (and not .exe files that execute in their own address space), you give your components maximum speed.

Active Server components (both those components intrinsic to ASP and those you create yourself) do not have a user interface. Indeed, if you forget this and design a server component with a msgbox or inputbox function, you will hang the server. This is because the program (namely, your errant server component) that is running is waiting for user input that never comes, because the user interface is invisible. In addition, server components need to support only three interfaces.

Finally, VBScript does not have a type library, so if you want to use variable types other than Variant, write a component. For example, if you want to extend the Active Server's mathematical power, you DO NOT want to use the Variant data type; it will slow your mathematical processing to a crawl. Instead, create a program in Visual Basic, and compile it as an in-process .dll file; register it with the operating system, and call it in your Active Server Pages.

4.5 ADO and *Server* Objects

Using ADO, perhaps more than any other Server Component highlights the symbiosis between Server Objects and Components. The main reason for this interdependence is that, because Web applications are based on individual HTML pages, these applications can be problematical for the Active Server developer. The problem arises because, while on one hand, the application moves freely and easily between several forms or instances of the same form (as you will see demonstrated later), at the same time, a stable substrate of data underlies the dynamic interplay of user interface. The developer's challenge is to keep this data substrate available to any and all pages that need it.

There are two specific issues facing the ADO developer- performance and reference. Performance issues arise because you cannot spare the bandwidth to make repeated calls to server

data stores for the same recordset (just because different pages need it). All pages should refer to the same fetched data.

Reference issues are similar to performance issues but different, because pages often will have to exchange data entered by the user; even more often, they will have to exchange data with each other (whether the data was fetched or entered). One or more pages needs to know what that data is and must be updated if that data is changed by another page in the application.

Figure 4.3 depicts these inter-relationships. A database contains information needed by an ASP application. The Active Server makes the call to fetch the data. If you use Session properties, discussed in the next section, you only have to fetch this data once; all the Active Server Pages will reference this copy of the data as necessary. If you do not use Session properties, then every time an .asp file needs the data, it has to require the data provider. Also note in the figure that data can "trickle down" from one page to several others. This data may affect and be affected by the data in the original database.

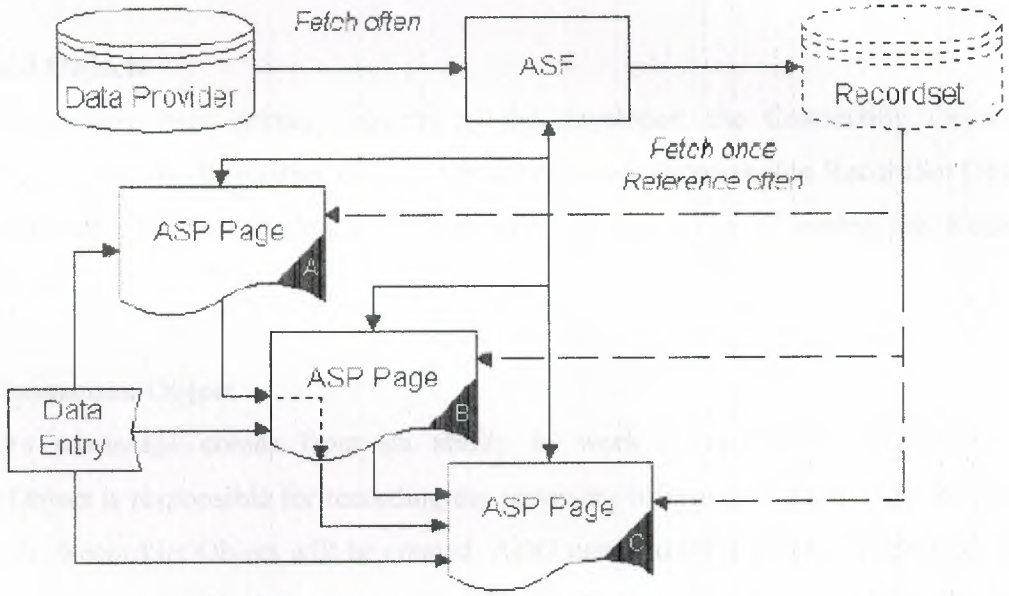

Figure 4.3 The problem of ADO programming centers in repeated calls to a data provider for data.

ASP has something for just this purpose, Session properties. Remember that VBScript is not yet able to reference a type library at design time and perform early binding. Therefore, all variables in VBScript are of type Variant. The performance issues that normally attend the exclusive use of variants usually are not issues in Active Server programming, because the client apps tend to remain thin. Intelligent design of server components enables the high-performance processing of variable data.

At any rate, Session properties can be created simply by setting them to a value. For example, a RecordSet Object can be created as a Session properties simply by stating the following:

Session("rstAuthors") = rstAuthors

4.6 The ADO Object Model

As mentioned, of all the Server Components that ship with ASP, ADO is the most complex and the one with the most long-term impact on the future of programming. In this chapter, we focus on the highest level of the object model and highlight some of the key properties and methods necessary to make ADO work at its most basic level. A key objective of this chapter is to stress that understanding the relationships between features of ADO is more important than an understanding of its parts.

4.6.1 Exposed Objects

ADO exposes three primary objects to the developer: the Connection Object, the Command Object, and the RecordSet Object. For all practical purposes, the RecordSet Object is the most important; the Connection and Command Objects serve to enable the RecordSet Object's creation.

4.6.2 The Connection Object

ADO's advantage comes from its ability to work in stateless environments. The Connection Object is responsible for recording the necessary information about the data provider from which the RecordSet Object will be created. ADO needs to inform the Windows NT server of the existence of an ODBC data provider by citing a Data Source Name (DSN). Recall that each DSN receives a name; the Connection Object refers to that name with its Open method and records the name in its ConnectionString property. The Open method also needs optional UserID and Password values, should the DSN require them.

Another feature of Connection Objects will be familiar to Access and SQL Server programmers. Like its predecessors, ADO can exploit the I/O efficiency of transactions using the BeginTrans, CommitTrans, and/or RollbackTrans methods. These methods-at least, in Access-are part of the Workspace Object, and you can see how it and the Connection Object exist for the

same reason. Think of the Connection Object as the telephone and circuits that enable you to call your mother; they don't do the talking, but without them, there's no conversation.

If the Connection Object is like the telephone, then its Open method is like placing a call, and its Execute method is like opening your mouth. Actually, there are several ways to create a RecordSet Object. Essentially, you can do it with the Connection, Command or even the RecordSet Object itself. For the moment, we remain focused on the Connection Object. Shows how you instantiate a Connection Object from the Database Access Component (identified by the ADO progid). You open this object by assigning it to the DSN named "Blotter". And you create a new recordset by invoking the Execute method.

Set objConn=Server.CreateObject("ADODB.Connection")
objConn.Open("Blotter")
Set objRst=objConn.Execute "tblBlotter"

As mentioned, the Execute method takes a given SQL command and interrogates the DSN with it. Success yields a RecordSet Object that is created with the VBScript Set command. Using this approach creates an implicit RecordSet Object, by the way. This means that ADO has given you a minimalist RecordSet Object; more important, it generates the least powerful cursor. Specifically, the resulting cursor is the row-wise, scroll-forward, read-only variety.

4.6.3 The Command Object

The Command Object provides the second way to create a RecordSet Object. This object also creates a minimal cursor, but it was designed to exploit a key concept in database management: passed parameters. Parameters are variables stored inside queries and stored procedures. Think of queries, and especially stored procedures, as mini-programs compiled by the data provider. Like normal programs, these objects can accept data at runtime that affect how the object behaves. Queries designed like this are called "parameterized queries." For example, if you want to list only certain records from a given table, you pass the name of the field and the value of interest to the parameterized query, and it filters out all other records from the resulting recordset.

Opening a Simple recordset with the Command Object

Set objCmd = Server.CreateObject(ADODB.Command)

```
objCmd.ActiveConnection ="intranet"
objCmd.CommandText="qryPhoneMessagesFor"
Set objRst = objCmd.Execute
```

Object: Refer to database objects such as queries, stored procedures, or table names; or use explicit SQL statements such as "SELECT * FROM tblBlotter". Referring to objects can yield dramatic improvements in performance because they can exploit all the processing power of the data provider. However, using a SQL statement makes your code self-documenting; i.e., you can tell exactly what your program is doing as it interacts with the data provider. In most cases, this advantage of self-documenting code is more than offset by the loss in performance; and besides, you can always explicitly document calls to database objects.

4.6.4 The RecordSet Object

When you use the RecordSet Object to *create* a recordset, you are using the CreateObject method of the Server Object to instantiate an explicit RecordSet Object. This means that you are responsible for specifying all the properties of the resulting recordset (unless you accept default values). Alternatively, you can create a recordset implicitly by using a Connection Object or the Command Object-but then you have no control over properties. If you need a dynamic cursor that is fully scrollable and permits batch updates, by using the RecordSet Object, that's exactly what you'll get.

At this point, you have made the call, connected with the other end, and are ready to start talking. In the same way that a conversation is full of words, recordsets are full of data. To fill the recordset with data, then, you don't use an Execute method; you use the Open method. Like the Execute method, the Open method in a Connection Object context is different from the Open method with the RecordSet Object. With the Connection Object, the Open method opens a channel permitting data to flow; with the RecordSet Object, the Open method fills a recordset with data.

The Code Necessary to Create a RecordSet Object

```
Set objRst = Server.CreateObject("ADODB.Recordset")
objRst.Source = "qryBlotterByDate"
objRst.ActiveConnection="Blotter"
objRst.Open
```

Each RecordSet Object contains a Fields collection of all the Field objects that are in the recordset. By manipulating the Fields collection, you change the structure of the underlying database table. That is, by referring to the Fields collection, you can construct SQL commands to update or otherwise modify the structure of the underlying tables at the data provider.

The real meat of the RecordSet Object, however, is in its methods and properties. To the extent that you do any serious database management using an HTML user interface, your ADO programs probably will use all those methods at one time or another.

4.7 Methods

4.7.1 Abandon

The Abandon method applies to Active Server Session Objects. Sessions are created as soon as a user opens an .asp file in a virtual directory of the Internet Information Server. This session stays open until one of two things happens. Either there is no activity from the user for 20 minutes (or the interval specified in the Timeout property of the Session Object) or an Abandon method is invoked. If you need access to any connected database before the session expires (for example, to back up the database), you need to *Abandon* it first.

4.7.2 CreateObject

The CreateObject method applies to the Server Object. This method creates instances of server components (such as TextStream) and ADO objects (such as Connection and RecordSet). The similarity with this method to its namesake in Visual Basic is that the Server Object must be part of the call; namely, Server.CreateObject(), not merely CreateObject().

4.7.3 Open

The Open method applies to Connection and RecordSet Objects. With the Connection Object, the method opens a connection-a channel of communication-to a server; specifically, to a Data Source Name. When invoked by a RecordSet Object, this method opens a cursor in a table in the DSN. The cursor is a current row-pointer within the recordset created with the Open method.

4.7.4 Requery

The Requery method applies to the RecordSet Object created with the Open method. Its function is to re-fire the query that populated the recordset, fetching the current-perhaps updated-values from the underlying database table.

4.7.5 Update

The Update method also is used by the RecordSet Object. It moves the data in the copy buffer to the RecordSet Object. Until this event occurs, the underlying table can have one value and the RecordSet Object another; after the Update method, they have the same data-unless something interfered with the routine processing of updates.

4.8 Properties

4.8.1 ActiveConnection

The ActiveConnection property tells ADO where the data is and how to access it. ActiveConnection functions like a telephone connection in that it enables communication but does not communicate directly. Some ADO connections are like station-to-station long distance. Others are more restricted, like a person-to-person call, limiting data access to only certain people with specific passwords.

4.8.2 CursorType

CursorType is an important property that applies to the RecordSet Object. It determines how hard the data provider has to work to make the records it stores available to your ADO program. The simplest cursor is a forward only, row-wise, read-only cursor. Other data providers can provide dynamic cursors that keep track of the status of underlying data.

4.8.3 LockType

The LockType property applies to RecordSet Objects and controls what results when the RecordSet Object executes its Open method. This property is important because it tells the data provider how to handle concurrency issues, should they arise.

4.8.4 Name

The Name property applies to the Field Object of the RecordSet Object.

4.8.5 Source

The Source property applies to RecordSet Objects. Usually, this is a text string of SQL commands to fetch data from the data provider. A shortcut is to use the name of the table alone; this is quicker than typing **SELECT * FROM tblBlotter**. Note, however, that if you want only a selected group of records or set of fields, you will have to use a SQL command, a query, or a SQL Server stored procedure.

4.9 Essential ADO

4.9.1 Preliminaries

To help you come to terms with recordsets, we take a brief detour into the world of relational database management. The comments that follow are for those who have had little use for databases until now or those who have used database systems but didn't have a need to get into the theory of database management. More experienced readers can safely skip this section.

4.9.2 Concurrency and Locking Issues

There is an adage as old as the Internet that goes something like "Information yearns to be free." In the database business world, there is a related truism: Data needs to be changed. The subject of this word play raises two serious problems for database developers.

On one hand, displayed data (especially when more than one row is being displayed) often needs to be up-to-date. This means that when someone adds or deletes a record, all other displays of data from that table need to reflect the change.

On the other hand, when the value of one or more fields of an existing record gets changed by more than one person and at the same time, there is a potential conflict, a collision of wills. The DBMS must be able to sense these collisions and manage them effectively.

Cursors play a part in both of these situations. More precisely, different cursor types play different roles in these different circumstances. Cursors have two primary flavors-static and dynamic. Static cursors, as their name suggests, can't see additions and deletions made by other users. For example, If a business application that works with only one record at a time is being built, it need not concern itself with the need to update the number of rows in a recordset. This is the kind of recordset that ADO creates by default. Other times, a dynamic cursor is needed. Dynamic cursors sense, on their own, when the number of rows or the content of fields changes.

The ASP developer also needs to address the issue of concurrency and the related issue of locking. Locking techniques fall into two categories: optimistic and pessimistic. Optimistic locking is relatively easy for the DBMS to implement, for it assumes that collisions and conflicts will be rare and doesn't activate locks until just before updating, and only if a conflict exists. Pessimistic locking assumes the opposite, and locks on data are required before processing a record can even begin. Again, not all DBMSs support both, and when some do, they don't give you a choice between the two.

4.9.3 Keys, Indexes, and Bookmarks

Relational Database Management Systems always work more efficiently if they can uniquely identify individual records. They also work more efficiently if certain fields are indexed; that is, put in order, such as last names in alphabetical order. Tables need keys to do both these things. Indexes sort records based on the values of these key fields; if these values are unique, they serve double duty-they sort and uniquely identify records. In addition, if a SQL Server table has a key, it can have a keyset cursor. Access isn't as picky; a table of two fields and no indexes does not return an error when a dynaset (the closest thing to a keyset cursor that Access has) is created and updated.

One more concept: bookmarks. Bookmarks are to cursors what cursors are to recordsets-they are placeholders. Some DBMSs (like FoxPro) keep track of record numbers. The Jet engine in Access and ADO do not. Instead, they rely on bookmarks to move the cursor to a previous location in a recordset. As you might guess, bookmarks are not supported by all cursors. Remember that dynamic cursors get updated when records are added or deleted. As a result, bookmarks aren't supported (in part, because the row that they used to represent may be gone). Only static and keyset cursors (and dynasets in Access) support bookmarks.

CHAPTER 5

DESCRIPTION OF ESTATE AGENT WEB PAGE

5.1 Introduction

My project is designing and programming a web page. This web page is about the estate agents.

5.2 Searching an Estate

Here we can search estates in details. The reason of putting this page is to search the estates in details.

Figure 5.1 Search an Estate Screen Shot.

5.3 Search Estate Agents

This page allows you to search estate agents by the first character of its name.

Figure 5.2 Lists of Estate Agents by Name Screen Shot.

5.4 Search Countries Estate Agents

This page allows searching the countries estate agents.

Figure 5.3 Search Countries Estate Agents Screen Shot

5.5 Rent / Sell an Agent

In this page estate agents must be login to add a new estate to database. If estate agent has not login the screen is coming(Figure 5.4).

Figure 5.4 Rent / Sell an Agent page Estate Agent did not logged in.

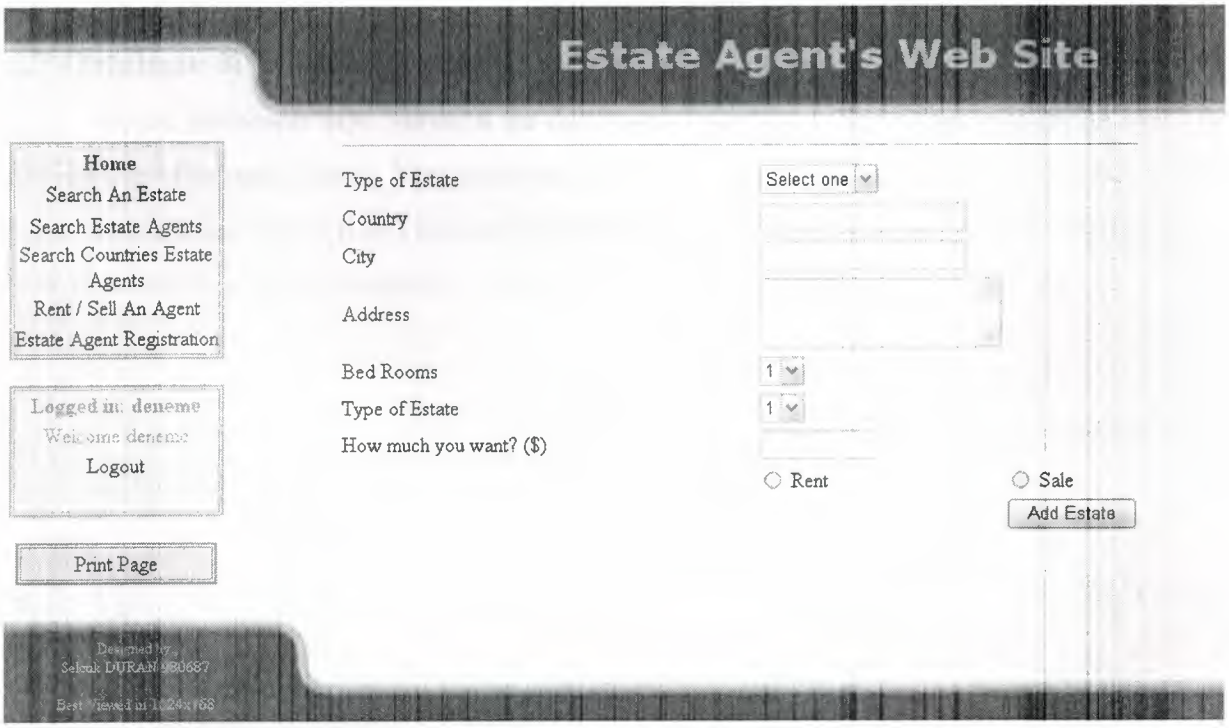

Figure 5.5 Rent / Sell an Agent page Estate Agent logged in.

5.6 Estate Agent Registration

Estate Agents can register to this web site. The registration data has to be looked up by the Webmaster and then must be send to Agent with username and password.

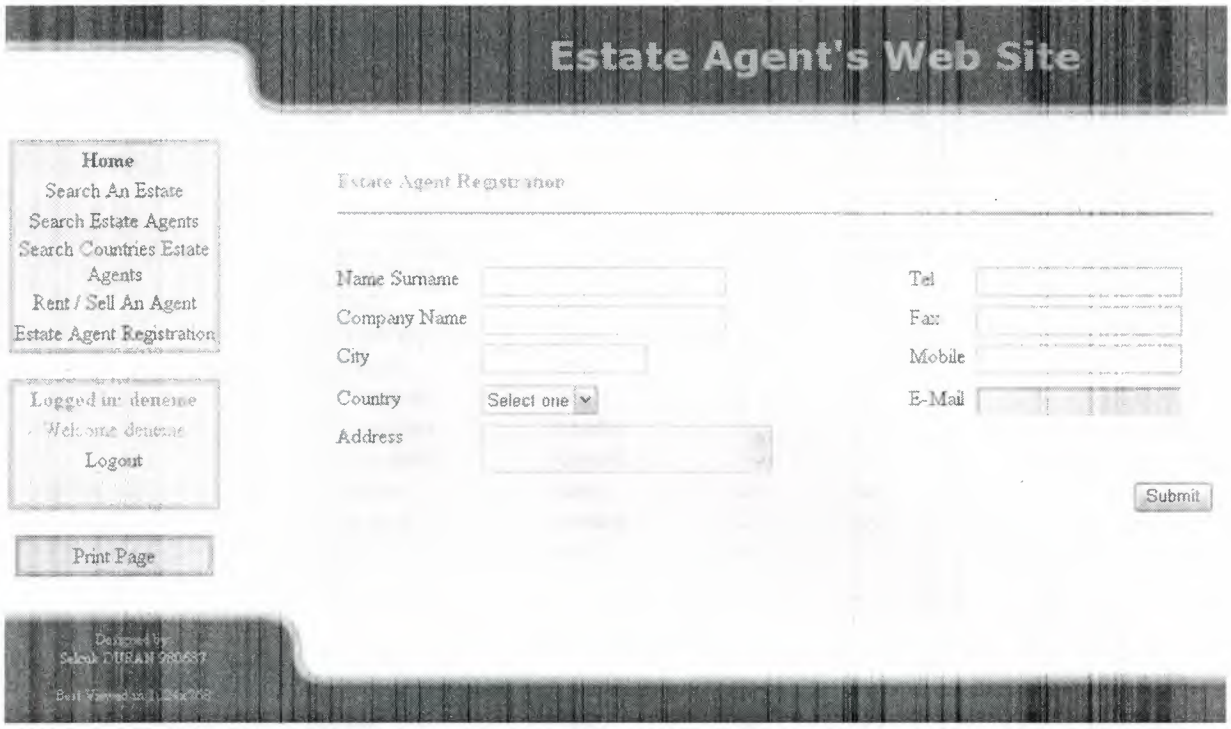

Figure 5.6 Estate Agent Registration Screen Shot.

5.7 Database & Tables

I used Microsoft SQL Server 8 for databases. I used SQL Server because in the web you can not use Microsoft Access. You must use Microsoft SQL Server or Oracle. I know that in web pages we must use one of this. I choose Microsoft SQL Server 8, because I know how to use it. I have three tables with no connection. The tables are shown in (Table 5.1 – 5.2 – 5.3).

Column Name	Data Type	Length	Allow Nulls
Name	char	25	✓
Coname	char	15	✓
Address	char	100	✓
City	char	20	✓
Country	char	20	✓
Tel	char	15	✓
Fax	char	15	✓
Mobile	char	15	✓
Email	char	50	✓
pri	int	4	

Table 5.1 Estate Agents

Column Name	Data Type	Length	Allow Nulls
Eaddress	char	100	✓
Ecity	char	20	✓
Ecountry	char	20	✓
Etype	char	20	✓
Erentsale	bit	1	✓
Enoofbed	smallint	2	✓
Enoofbath	smallint	2	✓
Eprice	char	12	✓
Erating	smallint	2	✓
pri	int	4	

Table 5.2 Estates

Column Name	Data Type	Length	Allow Nulls
UN	char	10	✓
PW	char	10	✓

Table 5.3 Estate Agent Usernames

INDEX.ASP

```
<!-- #include file="common.asp" -->
<%
set Conn = Server.CreateObject("ADODB.Connection")
Conn.ConnectionString = CommonConnectionString
Conn.Open
set RS = Server.CreateObject("ADODB.Recordset")
        if request.querystring("op")="li" then
                SQLStr = "SELECT * FROM db_login WHERE UN='" &
request.form("UserName") & "' AND PW='" & request.form("Password") & "'"
                RS.Open SQLStr, Conn, 1, 1
                if not RS.RecordCount=0 or not Rs.RecordCount=null then
                        session("Login") = "True"
                        session("UN") = request.form("UserName")
                end if
        end if
        if request.querystring("op")="lo" then
                session("UN") = null
                session("Login") = "False"
        end if
conn.close
set conn = nothing
%>
<html>
<head>
<title>Com 400 Graduation Project</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<style>
BODY{
                CURSOR: default;
                MARGIN-TOP: 0;
                MARGIN-LEFT: 0;
                MARGIN-RIGHT: 0;
```

```
            MARGIN-BOTTOM: 0;
        }
</style>
</head>
<body bgcolor="#000033" scroll=no oncontextmenu="return false" onselectstart="return
false" ondragstart="return false"  onLoad="MainFrame.location='main.asp';
window.status='Com 400 Graduation Project...';
window.parent.status_bar.style.visibility='hidden'"
onunload="window.parent.status_bar.style.visibility='visible'; this.focus()">
<table bgcolor="white" width="100%" height="84" border="0" cellpadding="0"
cellspacing="0" align="center">
        <tr>
   <td width="170" valign="top" bgcolor="#FFFFFF"> <IMG height=36
src="images/barleft.gif" width=170>
   </td>
                <td width="43" align="middle">
              <IMG height=84 src="images/barcenter.jpg" width=43></td>
   <td align="middle" valign="top" background="images/barright.gif">
    <p style="MARGIN-TOP: 20px; FONT-SIZE: 30px; COLOR: #ccaa00; FONT-
FAMILY: Verdana"><b>Estate
      Agent Program...</b></p>
                </td>
        </tr>
        <tr>
   <td valign="top" bgcolor="#FFFFFF"> <IMG height=1 src="images/spacer.gif" width=1>
    <table border="0" bordercolor="#888888" width="160" align="center" style="BORDER-
RIGHT: double; BORDER-TOP: double; BORDER-LEFT: double; BORDER-BOTTOM:
double" cellpadding="0" cellspacing="0">
      <tr onMouseOver="this.style.color='#0000CC';   this.style.cursor='hand';
window.status='Home Page'" onMouseOut="this.style.color=''; window.status='Com 400
Graduation Project...'" onClick="MainFrame.location='main.asp'">
        <td width="143" height="23" align="middle"
bgcolor="#ddffff"><strong>Home</strong>
        </td>
```

```
    </tr>
    <tr onMouseOver="this.style.color='steelblue'; this.style.cursor='hand';
window.status='Search An Estate'"
    onMouseOut="this.style.color=''; window.status='Com 400 Graduation Project...'"
    onClick="MainFrame.location='search_estate.asp'">
     <td height="23" align="middle" bgcolor="#eeffff">Search An Estate</td>
    </tr>
    <tr onMouseOver="this.style.color='steelblue'; this.style.cursor='hand';
window.status='Search Estate Agents'"
    onMouseOut="this.style.color=''; window.status='Com 400 Graduation Project...'"
    onClick="MainFrame.location='search_ea.asp'">
     <td height="23" align="middle" bgcolor="#eeffff">Search Estate Agents</td>
    </tr>
    <tr onMouseOver="this.style.color='steelblue'; this.style.cursor='hand';
window.status='Search Countries Estate Agents'"
    onMouseOut="this.style.color=''; window.status='Com 400 Graduation Project...'"
    onClick="MainFrame.location='search_ctry.asp'">
     <td height="23" align="middle" bgcolor="#eeffff">Search Countries Estate
      Agents</td>
    </tr>
<%
if session("Login")="True" then
%>
    <tr onMouseOver="this.style.color='steelblue'; this.style.cursor='hand';
window.status='Rent / Sell An Agent'"
    onMouseOut="this.style.color=''; window.status='Com 400 Graduation Project...'"
    onClick="MainFrame.location='rent_sell.asp'"
   >
     <td height="23" align="middle" bgcolor="#eeffff">Rent / Sell An Agent</td>
    </tr>
<%
else
%>
```

```
    <tr onMouseOver="this.style.color='steelblue'; this.style.cursor='hand';
window.status='Rent / Sell An Agent'"
    onMouseOut="this.style.color=''; window.status='Com 400 Graduation Project...'"
    onClick="MainFrame.location='rent_sell.asp'"
    >
     <td height="23" align="middle" bgcolor="#eeffff">Rent / Sell An Agent</td>
    </tr>
<%
end if
%>
    <tr onMouseOver="this.style.color='steelblue'; this.style.cursor='hand';
window.status='Estate Agent Registration'"
    onMouseOut="this.style.color=''; window.status='Com 400 Graduation Project...'"
    onClick="MainFrame.location='registration.asp'"
    >
     <td height="23" align="middle" bgcolor="#eeffff">Estate Agent Registration</td>
    </tr>
   </table>
   <br>
                              <table border="0" bordercolor="#888888" width="160"
align="center" style="BORDER-RIGHT: double; BORDER-TOP: double; BORDER-LEFT:
double; BORDER-BOTTOM: double" cellpadding="0" cellspacing="0">
<%
if not session("Login")="True" then
%>
<form name="Login" method="POST" action="index.asp?op=li">
    <tr>
     <td height="23" colspan="2" align="middle" bgcolor="#eeffff"><font
color="steelblue"><strong>Estate
       Agent Login</strong></font></td>
    </tr>
    <tr>
     <td width="75" height="23" align="middle" bgcolor="#eeffff"><div align="left">
       Username:</div></td>
```

```
      <td width="92" align="middle" bgcolor="#eeffff"><div align="right">
        <input name="UserName" type="text" value="username" size="10"
onFocus="this.value="">
      </div></td>
    </tr>
    <tr>
      <td height="11" align="middle" bgcolor="#eeffff"><div align="left">
        Password:</div>
      <div align="right"> </div></td>
      <td height="11" align="middle" bgcolor="#eeffff"><div align="right">
        <input name="Password" type="password" value="password" size="8"
onFocus="this.value="">
      </div></td>
    </tr>
    <tr>
      <td height="12" colspan="2" align="middle" bgcolor="#eeffff"><div align="right">
        <input type="submit" name="Submit" value="Login">
      </div></td>
    </tr>
</form>
<%
else
%>
    <tr>
      <td height="23" colspan="2" align="middle" bgcolor="#eeffff"><font
color="steelblue"><strong>Logged in: <%=session("UN")%></strong></font></td>
    </tr>
    <tr onMouseOut="this.style.color=''; window.status='Com 400 Graduation Project...'">
      <td height="23" align="middle" bgcolor="#eeffff"><font color="steelblue">Welcome
<%=session("UN")%></font></td>
    </tr>
    <tr onMouseOver="this.style.color='steelblue'; this.style.cursor='hand';
window.status='Search Estate Agents'"
    onMouseOut="this.style.color=''; window.status='Com 400 Graduation Project...'"
```

```
      onClick="document.location='index.asp?op=lo'">
        <td height="23" align="middle" bgcolor="#eeffff">Logout</td>
      </tr>
      <tr onMouseOut="this.style.color=''; window.status='Com 400 Graduation Project...'">
        <td height="23" align="middle" bgcolor="#eeffff"> </td>
      </tr>
<%
end if
%>
    </table>
    <br>
                        <table border="0" bordercolor="#555555" width="150"
align="center" style="BORDER-RIGHT: double; BORDER-TOP: double; BORDER-LEFT:
double; BORDER-BOTTOM: double" cellpadding="0" cellspacing="0">
      <tr onMouseOver="this.style.color='steelblue'; this.style.cursor='hand';
window.status='Send Document To The Printer'" onMouseOut="this.style.color='';
window.status='Com 400 Graduation Project'"
onClick="MainFrame.focus();MainFrame.print()"><td height="23" align="middle"
bgcolor="#ddeaff">Print Page</td></tr>
                        </table>
                        <div id="status_bar">
                            <table border=0 cellspacing="0" align=center>
                                <tr>
                                    <td align=middle>
                                        <IMG height=20
src="images/status.gif" width=120>
                                    </td>
                                </tr>
                            </table>
                        </div>
            </td>
    <td bgcolor="#FFFFFF"> <IMG height=1 src="images/spacer.gif" width=1>
    </td>
```

```
  <td valign="top" bgcolor="#FFFFFF"> <IMG height=1 src="images/spacer.gif" width=1>
   <table border="0" width="758" align="center">
    <tr>
     <td height="308" bgcolor="#FFFFFF">
      <iframe frameborder="0" width="100%" height="100%" name="MainFrame"
border="0" scrolling="auto"></iframe>
     </td>
    </tr>
   </table>
  </td>
      </tr>
</table>
<table bgcolor=white width="100%" cellpadding="0" cellspacing="0">
      <tr>
            <td valign="top" width="200" background="images/barleftbot.gif">
                  <center>
                        <p style="FONT-SIZE: 11px; COLOR: #aaaaaa">
                              <br>
      Designed by:<br>
      <a style="COLOR: #dddddd">Selcuk DURAN 980687</a><br>
      <br>
      Best Viewed in 1024x768</p>
                  </center>
            </td>
            <td width="43"><IMG height=79 src="images/barcenterbot.jpg"
width=43></td>
  <td valign="bottom" align="right"><IMG height=32 src="images/barrightbot.gif"
width=100%></td>
      </tr>
</table>
</body>
</html>
```

ACTION.ASP

```
<body onload="window.parent.status_bar.style.visibility='hidden'"
onunload="window.parent.status_bar.style.visibility='visible'"
scroll=auto oncontextmenu="return false" ondragstart="return false">
<!-- #include file = "common.asp" -->
<%
dim Conn
set Conn = Server.CreateObject("ADODB.Connection")
Conn.ConnectionString = CommonConnectionString
Conn.Open
dim RS
set RS = Server.CreateObject("ADODB.Recordset")
%>
<%
select case (request.QueryString("op"))
        case "qs"
                if request.Form("qrycat") = 1 then SQLStr = "SELECT * FROM db_Estates
WHERE EType LIKE '%" & request.Form("qrystr") & "%'"
                if request.Form("qrycat") = 2 then SQLStr = "SELECT * FROM db_eagents
WHERE Name LIKE '%" & request.Form("qrystr") & "%' OR Coname LIKE '%" &
request.Form("qrystr") & "%'"
                if request.Form("qrycat") = 3 then SQLStr = "SELECT * FROM db_Estates
WHERE Ecountry LIKE '%" & request.Form("qrystr") & "%'"
                if request.Form("qrycat") = 3 then SQLStr2 = "SELECT * FROM db_eagents
WHERE Country LIKE '%" & request.Form("qrystr") & "%'"
                RS.Open SQLStr, Conn, 1, 1
                select case (request.Form("qrycat"))
                        case 1
                                if rs.recordcount <> 0 then
                                        response.Write("<table border='0' width='80%'
cellspacing='0' cellpadding='0'>")
                                        response.Write("<tr>")
```

```
response.Write("<th>Type</th><th>Rent/Sale</th><th>City</th><th>Country</th><
th>Rating</th><th>Price</th></tr>")
                    do while not RS.EOF
                        if RS("Erentsale") = True then RorS = "for Rent"
                        if RS("Erentsale") = False then RorS = "for Sale"
                        CreateResult = "<tr
onmouseover=""this.style.background='lightsteelblue';this.style.cursor='hand'""
onmouseout=""this.style.background="""" onclick=""window.open('result.asp?wh=e&ID=" &
RS("Pri") &
"',null,'top=200,left=300,width=200,height=250,resizable=no,scrollbar=no,menubar=no,toolb
ar=no,statusbar=no')""><td>"
                        CreateResult = CreateResult & RS("Etype") &
"</td><td>"
                        CreateResult = CreateResult & RorS &
"</td><td>"
                        CreateResult = CreateResult & RS("Ecity") &
"</td><td>"
                        CreateResult = CreateResult & RS("Ecountry") &
"</td><td>"
                        CreateResult = CreateResult & RS("Erating") &
"</td><td>"
                        CreateResult = CreateResult & RS("Eprice") &
"</td></tr>"
                        response.Write(CreateResult)
                        RS.MoveNext
                    loop
                    response.Write("</table>")
                else
                    response.Write("No Estates Found In Your Search...")
                end if
            case 2
                if rs.recordcount <> 0 then
```

```
                    response.Write("<table border='0' width='80%'
cellspacing='0' cellpadding='0'>")
                    response.Write("<tr>")
        response.Write("<th>Name</th><th>Company</th><th>City</th><th>Country</th>
<th>Phone</th><th>E-Mail</th></tr>")
                    do while not RS.EOF
                        CreateResult = "<tr
onmouseover=""this.style.background='lightsteelblue';this.style.cursor='hand'""
onmouseout=""this.style.background="""" onclick=""window.open('result.asp?wh=ea&ID="
& RS("Pri") &
"',null,'top=200,left=300,width=200,height=250,resizable=no,scrollbar=no,menubar=no,toolb
ar=no,statusbar=no')""><td>"
                        CreateResult = CreateResult & RS("Name") &
"</td><td>"
                        CreateResult = CreateResult & RS("Coname") &
"</td><td>"
                        CreateResult = CreateResult & RS("City") &
"</td><td>"
                        CreateResult = CreateResult & RS("Country") &
"</td><td>"
                        CreateResult = CreateResult & RS("Tel") &
"</td><td>"
                        CreateResult = CreateResult & RS("Email") &
"</td></tr>"
                        response.Write(CreateResult)
                        RS.MoveNext
                    loop
                    response.Write("</table>")
                else
                    response.Write("No Estate Agent Found In Your
Search...")
                end if
            case 3
            response.Write("<p style='color:darkred'><b>Estates:</b></p><br>")
```

```
if rs.recordcount <> 0 then
        response.Write("<table border='0' width='80%'
cellspacing='0' cellpadding='0'>")
        response.Write("<tr>")
    response.Write("<th>Type</th><th>Rent/Sale</th><th>City</th><th>Country</th><
th>Rating</th><th>Price</th></tr>")
        do while not RS.EOF
            if RS("Erentsale") = True then RorS = "for Rent"
            if RS("Erentsale") = False then RorS = "for Sale"
            CreateResult = "<tr
onmouseover=""this.style.background='lightsteelblue';this.style.cursor='hand'""
onmouseout=""this.style.background=""" onclick=""window.open('result.asp?wh=e&ID=" &
RS("Pri") &
"',null,'top=200,left=300,width=200,height=250,resizable=no,scrollbar=no,menubar=no,toolb
ar=no,statusbar=no')""><td>"
            CreateResult = CreateResult & RS("Etype") &
"</td><td>"
            CreateResult = CreateResult & RorS &
"</td><td>"
            CreateResult = CreateResult & RS("Ecity") &
"</td><td>"
            CreateResult = CreateResult & RS("Ecountry") &
"</td><td>"
            CreateResult = CreateResult & RS("Erating") &
"</td><td>"
            CreateResult = CreateResult & RS("Eprice") &
"</td></tr>"
            response.Write(CreateResult)
            RS.MoveNext
        loop
        response.Write("</table>")
    else
        response.Write("No Estates Found In Your Search...")
    end if
```

```
dim RS2
set RS2 = Server.CreateObject("ADODB.Recordset")
RS2.Open SQLStr2, Conn, 1, 1
response.Write("<br><br><hr><p style='color:darkred'><b>Estate
Agents:</b></p><br>")
    if rs2.recordcount <> 0 then
        response.Write("<table border='0' width='80%'
cellspacing='0' cellpadding='0'>")
        response.Write("<tr>")
    response.Write("<th>Name</th><th>Company</th><th>City</th><th>Country</th>
<th>Phone</th><th>E-Mail</th></tr>")
        do while not RS2.EOF
            CreateResult = "<tr
onmouseover=""this.style.background='lightsteelblue';this.style.cursor='hand'""
onmouseout=""this.style.background="""" onclick=""window.open('result.asp?wh=ea&ID="
& RS2("Pri") &
"',null,'top=200,left=300,width=200,height=250,resizable=no,scrollbar=no,menubar=no,toolb
ar=no,statusbar=no')""><td>"
            CreateResult = CreateResult & RS2("Name") &
"</td><td>"
            CreateResult = CreateResult & RS2("Coname")
& "</td><td>"
            CreateResult = CreateResult & RS2("City") &
"</td><td>"
            CreateResult = CreateResult & RS2("Country")
& "</td><td>"
            CreateResult = CreateResult & RS2("Tel") &
"</td><td>"
            CreateResult = CreateResult & RS2("Email") &
"</td></tr>"
            response.Write(CreateResult)
            RS2.MoveNext
        loop
        response.Write("</table>")
```

```
                    else
                        response.Write("No Estate Agents Found In Your
Search...")
                    end if
                    RS2.close
                    set RS2 = nothing
            end select
        case "se"
            SQLStr = "SELECT * FROM db_Estates WHERE "
            SQLStr = SQLStr & "Ecountry='" & Convert(1,request.Form("frmCountry"),0)
& "' "
            SQLStr = SQLStr & "AND Enoofbed=" & request.Form("frmBedRoom") & "
"
            SQLStr = SQLStr & "AND Enoofbath=" & request.Form("frmBathRoom") &
" "
            SQLStr = SQLStr & "AND Ecity='" & request.Form("frmCity") & "' "
            SQLStr = SQLStr & "AND Etype='" & Convert(2,request.Form("frmType"),0)
& "' "
            SQLStr = SQLStr & "AND Erentsale=" & request.Form("frmRorS") & " "
            SQLStr = SQLStr & "AND Eprice>=" & request.Form("frmPrice1") & " "
            SQLStr = SQLStr & "AND Eprice<=" & request.Form("frmPrice2")
            RS.Open SQLStr, Conn, 1, 1
            if rs.recordcount <> 0 then
                response.Write("<table border='0' width='80%' cellspacing='0'
cellpadding='0'>")
                response.Write("<tr>")
        response.Write("<th>Type</th><th>Rent/Sale</th><th>City</th><th>Country</th><
th>Rating</th><th>Price</th></tr>")
                do while not RS.EOF
                    if RS("Erentsale") = True then RorS = "for Rent"
                    if RS("Erentsale") = False then RorS = "for Sale"
                CreateResult = "<tr
onmouseover=""this.style.background='lightsteelblue';this.style.cursor='hand'""
onmouseout=""this.style.background="" onclick=""window.open('result.asp?wh=e&ID=" &
```

```
RS("Pri") &
"',null,'top=200,left=300,width=200,height=250,resizable=no,scrollbar=no,menubar=no,toolb
ar=no,statusbar=no')""><td>"
                    CreateResult = CreateResult & RS("Etype") & "</td><td>"
                    CreateResult = CreateResult & RorS & "</td><td>"
                    CreateResult = CreateResult & RS("Ecity") & "</td><td>"
                    CreateResult = CreateResult & RS("Ecountry") & "</td><td>"
                    CreateResult = CreateResult & RS("Erating") & "</td><td>"
                    CreateResult = CreateResult & RS("Eprice") & "</td></tr>"
                    response.Write(CreateResult)
                    RS.MoveNext
                loop
                response.Write("</table>")
            else
                response.Write("Your Detailed Search Could Not Be Found...")
            end if
        case "sa"
            SQLStr = "SELECT * FROM db_eagents WHERE coname LIKE '" &
request.QueryString("agent") & "%'"
            rs.open sqlstr, conn, 1, 1
            if rs.recordcount <> 0 then
                response.Write("<table border='0' width='80%' cellspacing='0'
cellpadding='0'>")
                response.Write("<tr>")
        response.Write("<th>Name</th><th>Company</th><th>City</th><th>Country</th>
<th>Phone</th><th>E-Mail</th></tr>")
                do while not RS.EOF
                    CreateResult = "<tr
onmouseover=""this.style.background='lightsteelblue';this.style.cursor='hand'""
onmouseout=""this.style.background="""" onclick=""window.open('result.asp?wh=ea&ID="
& RS("Pri") &
"',null,'top=200,left=300,width=200,height=250,resizable=no,scrollbar=no,menubar=no,toolb
ar=no,statusbar=no')""><td>"
                    CreateResult = CreateResult & RS("Name") & "</td><td>"
```

```
                    CreateResult = CreateResult & RS("Coname") & "</td><td>"
                    CreateResult = CreateResult & RS("City") & "</td><td>"
                    CreateResult = CreateResult & RS("Country") & "</td><td>"
                    CreateResult = CreateResult & RS("Tel") & "</td><td>"
                    CreateResult = CreateResult & RS("Email") & "</td></tr>"
                    response.Write(CreateResult)
                    RS.MoveNext
                loop
                response.Write("</table>")
            else
                response.Write("There is no Estate Agent Starting With
...<strong><font size='+3'>"& request.QueryString("agent")&"</font></strong>")
            end if
        case "sc"
            SQLStr = "SELECT * FROM db_eagents WHERE Country LIKE '%" &
request.Form("frmCountry") & "%'"
            RS.Open SQLStr, Conn, 1, 1
            if rs.recordcount <> 0 then
                response.Write("<table border='0' width='80%' cellspacing='0'
cellpadding='0'>")
                response.Write("<tr>")
        response.Write("<th>Name</th><th>Company</th><th>City</th><th>Country</th>
<th>Phone</th><th>E-Mail</th></tr>")
                do while not RS.EOF
                    CreateResult = "<tr
onmouseover=""this.style.background='lightsteelblue';this.style.cursor='hand'""
onmouseout=""this.style.background="""" onclick=""window.open('result.asp?wh=ea&ID="
& RS("Pri") &
"',null,'top=200,left=300,width=200,height=250,resizable=no,scrollbar=no,menubar=no,toolb
ar=no,statusbar=no')""><td>"
                    CreateResult = CreateResult & RS("Name") & "</td><td>"
                    CreateResult = CreateResult & RS("Coname") & "</td><td>"
                    CreateResult = CreateResult & RS("City") & "</td><td>"
                    CreateResult = CreateResult & RS("Country") & "</td><td>"
```

```
                    CreateResult = CreateResult & RS("Tel") & "</td><td>"
                    CreateResult = CreateResult & RS("Email") & "</td></tr>"
                    response.Write(CreateResult)
                    RS.MoveNext
                loop
                response.Write("</table>")
            else
                response.Write("There is no Estate Agent In  ...<strong><font
size='+1'>"& request.form("frmCountry")&"</font></strong>")
            end if
        case "rs"
            if Request.Form("frmRentSell")="" or Request.Form("frmRentSell")=null then
                Response.Write("Please select your estate type: Rent or Sale?<br><a
href='javascript:history.back(1)'>Back</a>")
                Response.End
            end if
            SQLStr = "INSERT INTO db_Estates ("
            SQLStr = SQLStr &
"Eaddress,Ecity,Ecountry,Etype,Erentsale,Enoofbed,Enoofbath,Eprice) "
            SQLStr = SQLStr & "VALUES ('"
            SQLStr = SQLStr & request.Form("frmAddr") & "','"
            SQLStr = SQLStr & request.Form("frmCity") & "','"
            SQLStr = SQLStr & request.Form("frmCountry") & "','"
            SQLStr = SQLStr & request.Form("frmType") & "',"
            SQLStr = SQLStr & request.Form("frmRentSell") & ","
            SQLStr = SQLStr & request.Form("frmBedRoom") & ","
            SQLStr = SQLStr & request.Form("frmBathRoom") & ","
            SQLStr = SQLStr & request.Form("frmPrice") & ")"
            Conn.execute(SQLStr)
        case "ar"
            SQLStr = "INSERT INTO db_eagents ("
            SQLStr = SQLStr &
"Name,Coname,Address,City,Country,Tel,Fax,Mobile,Email) "
            SQLStr = SQLStr & "VALUES ('"
```

```
                SQLStr = SQLStr & request.Form("frmName") & "','"
                SQLStr = SQLStr & request.Form("frmCoName") & "','"
                SQLStr = SQLStr & request.Form("frmAddress") & "','"
                SQLStr = SQLStr & request.Form("frmCity") & "','"
                SQLStr = SQLStr & Convert(1,request.Form("frmCountry"),0) & "','"
                SQLStr = SQLStr & request.Form("frmTel") & "','"
                SQLStr = SQLStr & request.Form("frmFax") & "','"
                SQLStr = SQLStr & request.Form("frmMobile") & "','"
                SQLStr = SQLStr & request.Form("frmEmail") & "')"
                Conn.Execute (SQLStr)
        case "li"
                SQLStr = "SELECT * FROM db_login WHERE UN='" &
request.form("UserName") & "' AND PW='" & request.form("Password") & "'"
                RS.Open SQLStr, Conn, 1, 1
                if not RS.RecordCount=0 or RS.RecordCount=null then
                        session("Login") = True
                        session("UN") = request.form("UserName")
                end if
        case "lo"
                session("UN") = null
                session("Login") = False
end select
%>
<%
Conn.Close
set Conn = nothing
%>
</body>
```

COMMON.ASP

```
<%
CommonConnectionString = "Provider=SQLOLEDB.1;User
ID=sa;Password=ratna57b;Initial Catalog=estate agent;Data Source=PC"
```

```
function Convert(which,what,wh)
        select case which
                'country converting
                case 1
                        if wh=1 then
                                select case lcase(trim(what))
                                        case "turkey"
                                                Convert=1
                                        case "cyprus"
                                                Convert=2
                                end select
                        else
                                select case what
                                        case 1
                                                Convert="Turkey"
                                        case 2
                                                Convert="Cyprus"
                                end select
                        end if
                'type converting
                case 2
                        if wh=1 then
                                select case what
                                        case "villa"
                                                Convert=1
                                        case "flat"
                                                Convert=2
                                        case "bungalow"
                                                Convert=3
                                end select
                        else
                                select case lcase(trim(what))
                                        case 1
                                                Convert="villa"
```

```
                    case 2
                        Convert="flat"
                    case 3
                        Convert="bungalow"
                end select
            end if
        end select
end function
%>
```

MAIN.ASP

```
<!-- #include file = "common.asp" -->
<html>
<head>
<body onload="window.parent.status_bar.style.visibility='hidden'"
onunload="window.parent.status_bar.style.visibility='visible'"
scroll=no oncontextmenu="return false" ondragstart="return false">
<title></title>
<table width="750" height="256" border="0" bgcolor="#FFFFFF">
 <tr>
  <td width="787" height="252" colspan="5" align="center" valign="top">
   <table width="100%" height="12%">
    <tr>
     <td height="26"> <p align="center">  </p></td>
     <td height="26" colspan="2"> </td>
     <td height="26">Estate Agents Login</td>
    </tr>
     <form name="form1" method="post" action="action.asp?op=qs">
    <tr>
     <td height="21" align="center"> </td>
     <td colspan="2" bgcolor="#FFFFFF"><font color="#000000">Quick Search
      <input type="text" name="qrystr">
      <select name="qrycat">
```

```
          <option value="0" selected>Select one</option>
          <option value="1">Estate</option>
          <option value="2">Estate Agent</option>
          <option value="3">Country</option>
        </select>
<script language="JavaScript">
        function DontGo()
                {
                        if (form1.qrycat.value=='0')
                                {
                                        alert("Sec birini");
                                }
                        else
                                {
                                        form1.submit();
                                }
                }
</script>
        <input name="go" type="button" id="go" value="Go" onClick="DontGo()">
        </font></td>
       <td><a href="registration.asp">New Estate Agent</a></td>
      </tr>  </form>
      <tr>
       <td height="21" align="center"> </td>
       <td width="40%" bgcolor="#FFFFFF"> <p> </p></td>
       <td width="28%" bgcolor="#FFFFFF"> </td>
       <td> </td>
      </tr>
      <tr>
       <td height="21" align="center"> </td>
       <td colspan="2" bgcolor="#FFFFFF"> </td>
       <td> </td>
      </tr>
      <tr>
```

```
      <td height="21" align="center"> </td>
      <td colspan="2" bgcolor="#FFFFFF"> <p> </p></td>
      <td> </td>
     </tr>
     <tr>
      <td height="21" align="center"> </td>
      <td colspan="2" bgcolor="#FFFFFF"> </td>
      <td> </td>
     </tr>
     <tr>
      <td width="11%" height="21" align="center"> </td>
      <td colspan="2" bgcolor="#FFFFFF"> <p> </p></td>
      <td width="21%"> </td>
     </tr>
    </table>
    <div align="center"></div></td>
  </tr>
</table>
</body>
</html>
```

SEARCH_ESTATE.ASP

```
<%@LANGUAGE="VBSCRIPT" CODEPAGE="1252"%>
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body bgcolor="#FFFFFF" onload="window.parent.status_bar.style.visibility='hidden'"
onunload="window.parent.status_bar.style.visibility='visible'"
scroll=auto oncontextmenu="return false" ondragstart="return false">
<form name="frmSE" method="post" action="action.asp?op=se">
```

```
<table width="600" height="209" border="0">
  <tr>
   <td height="21" colspan="2"><strong><font color="steelblue">Search An
Estate</font></strong></td>
   <td width="22%"> </td>
   <td width="23%"> </td>
  </tr>
  <tr>
   <td height="21" colspan="4"><hr width="100%" size="2" color="steelblue"></td>
  </tr>
  <tr>
   <td width="25%" height="21">Country</td>
   <td width="30%">
              <select name="frmCountry">
     <option value="0" selected>Select one</option>
     <option value="1">Turkey</option>
     <option value="2">Cyprus</option>
    </select>
       </td>
   <td>No of Bedrooms</td>
   <td>
            <select name="frmBedRoom">
     <option value="0" selected>Select one</option>
     <option value="1">1</option>
     <option value="2">2</option>
     <option value="3">3</option>
     <option value="4">4</option>
     <option value="5">5</option>
    </select>
      </td>
  </tr>
  <tr>
   <td height="21">City</td>
   <td><input name="frmCity" type="text" size="15" maxlength="20"></td>
```

```
<td>No of Bathrooms</td>
<td>
        <select name="frmBathRoom">
  <option value="0" selected>Select one</option>
  <option value="1">1</option>
  <option value="2">2</option>
  <option value="3">3</option>
 </select>
    </td>
</tr>
<tr>
<td height="21">Type of Estate</td>
<td>
        <select name="frmType">
  <option value="0" selected>Select one</option>
  <option value="1">Villa</option>
  <option value="2">Mustakil</option>
  <option value="3">Flat</option>
  <option value="4">dublex</option>
  <option value="5">Apartment</option>
 </select>
    </td>
<td> </td>
<td> </td>
</tr>
<tr>
<td height="21"> </td>
<td> </td>
<td> </td>
<td> </td>
</tr>
<tr>
<td height="21">rent / sale</td>
<td>
```

```
            <select name="frmRorS">
        <option value="X" selected>select one</option>
        <option value="1">rent</option>
        <option value="0">for sale</option>
      </select>
          </td>
    <td>Price level</td>
    <td>
                <input name="frmPrice1" type="text" size="3"> - <input type="text"
name="frmPrice2" size="3">
          </td>
   </tr>
   <tr>
    <td height="21"> </td>
    <td> </td>
    <td>Rating</td>
    <td> </td>
   </tr>
   <tr>
    <td height="21"> </td>
    <td> </td>
    <td> </td>
    <td align="right"> <input type="submit" name="Submit" value="Search"> </td>
   </tr>
  </table>
</form>
</body>
</html>
```

SEARCH_EA.ASP

```
<%@LANGUAGE="VBSCRIPT" CODEPAGE="1252"%>
<html>
<head>
```

```
<title></title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body bgcolor="#FFFFFF" onload="window.parent.status_bar.style.visibility='hidden'"
onunload="window.parent.status_bar.style.visibility='visible'"
scroll=auto oncontextmenu="return false" onselectstart="return false" ondragstart="return
false">
<table width="600" height="214" border="0">
 <tr>
  <td height="21" colspan="2"> <div align="left"><strong><font color="steelblue">List
     of Estate Agents by Name </font></strong></div></td>
  <td colspan="2"><div align="right"></div></td>
 </tr>
 <tr bgcolor="bbffff">
  <td height="21" colspan="4"> <div align="center"><strong><a
href="action.asp?op=sa&agent=a"><font face="Times New Roman, Times,
serif">A</font></a><font face="Times New Roman, Times, serif">
     | <a href="action.asp?op=sa&agent=b">B</a> | <a
href="action.asp?op=sa&agent=c">C</a>
     | <a href="action.asp?op=sa&agent=d">D</a> | <a
href="action.asp?op=sa&agent=e">E</a>
     | <a href="action.asp?op=sa&agent=f">F</a> | <a
href="action.asp?op=sa&agent=g">G</a>
     | <a href="action.asp?op=sa&agent=h">H</a> | <a
href="action.asp?op=sa&agent=i">I</a>
     | <a href="action.asp?op=sa&agent=j">J</a> | <a
href="action.asp?op=sa&agent=k">K</a>
     | <a href="action.asp?op=sa&agent=l">L</a> | <a
href="action.asp?op=sa&agent=m">M</a>
     | <a href="action.asp?op=sa&agent=n">N</a> | <a
href="action.asp?op=sa&agent=o">O</a>
     | <a href="action.asp?op=sa&agent=p">P</a> | <a
href="action.asp?op=sa&agent=q">Q</a>
```

```
      | <a href="action.asp?op=sa&agent=r">R</a> | <a
href="action.asp?op=sa&agent=s">S</a>
      | <a href="action.asp?op=sa&agent=t">T</a> | <a
href="action.asp?op=sa&agent=u">U</a>
      | <a href="action.asp?op=sa&agent=v">V</a> | <a
href="action.asp?op=sa&agent=w">W</a>
      | <a href="action.asp?op=sa&agent=y">Y</a> | <a
href="action.asp?op=sa&agent=x">X</a>
      | <a href="action.asp?op=sa&agent=z">Z</a></font></strong></div></td>
  </tr>
  <tr>
   <td height="21" colspan="4"><hr align="center" size="2" color="steelblue">
   </td>
  </tr>
  <tr>
   <td height="21" colspan="4"><div align="right"></div></td>
  </tr>
  <tr>
   <td width="23%" height="21"> </td>
   <td width="32%"> </td>
   <td width="22%"> </td>
   <td width="23%"> </td>
  </tr>
  <tr>
   <td height="21"> </td>
   <td> </td>
   <td> </td>
   <td> </td>
  </tr>
  <tr>
   <td height="21"> </td>
   <td> </td>
   <td> </td>
   <td> </td>
```

```
</tr>
<tr>
  <td height="21"> </td>
  <td> </td>
  <td> </td>
  <td> </td>
</tr>
<tr>
  <td height="26"> </td>
  <td> </td>
  <td colspan="2"><div align="right"><a href="search_ctry.asp">Search Countries
    Estate Agents </a></div></td>
</tr>
</table>
</body>
</html>
```

SEARCH_CTRY.ASP

```
<%@LANGUAGE="VBSCRIPT" CODEPAGE="1252"%>
<html>
<head>
<title></title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body bgcolor="#FFFFFF" onload="window.parent.status_bar.style.visibility='hidden'"
onunload="window.parent.status_bar.style.visibility='visible'"
scroll=no oncontextmenu="return false" ondragstart="return false">
<table width="600" height="219" border="0">
  <tr>
    <td height="21" colspan="2"> <div align="left"><strong><font color="steelblue">Search
      Countries Estate Agents</font></strong></div></td>
    <td colspan="2"><div align="right"></div></td>
```

```
</tr>
<tr>
 <td height="21" colspan="4"><div align="left">
   <hr width="100%" size="2" color="steelblue">
  </div></td>
</tr>
<form name="form1" method="post" action="action.asp?op=sc">
 <tr>
  <td>Select a Country</td>
  <td>
      <input type="text" name="frmCountry">
       </td>
      </tr>
      <tr>
       <td align="center" colspan="2"><input type="submit" name="Button1"
value="Go"></td>
  </tr>
</form>
<tr>
  <td height="21"> </td>
  <td> </td>
  <td> </td>
  <td> </td>
</tr>
<tr>
  <td height="21"> </td>
  <td> </td>
  <td> </td>
  <td> </td>
</tr>
<tr>
  <td height="21"> </td>
  <td> </td>
  <td width="21%"> </td>
```

```
  <td width="24%"> </td>
 </tr>
 <tr>
  <td height="21"> </td>
  <td> </td>
  <td> </td>
  <td> </td>
 </tr>
 <tr>
  <td height="21"> </td>
  <td> </td>
  <td> </td>
  <td> </td>
 </tr>
 <tr>
  <td height="26"> </td>
  <td> </td>
  <td colspan="2"><div align="right"><a href="search_ea.asp">List of Estate
    Agents by Name </a></div></td>
 </tr>
</table>
</body>
</html>
```

RENT_SELL.ASP

```
<%@LANGUAGE="VBSCRIPT" CODEPAGE="1252"%>
<html>
<head>
<title></title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body bgcolor="#FFFFFF" onload="window.parent.status_bar.style.visibility='hidden'"
onunload="window.parent.status_bar.style.visibility='visible'"
```

```
scroll=auto oncontextmenu="return false" ondragstart="return false">
<%
if not session("Login")="True" then
%>
<table border="0" bordercolor="#888888" width="160" align="center" style="BORDER-
RIGHT: double; BORDER-TOP: double; BORDER-LEFT: double; BORDER-BOTTOM:
double" cellpadding="0" cellspacing="0">
        <form name="Login" method="POST" action="index.asp?op=li" target="_parent">
            <tr>
             <td height="23" colspan="2" align="middle" bgcolor="#eeffff"><font
color="steelblue"><strong>Estate
               Agent Login</strong></font></td>
            </tr>
            <tr>
             <td width="75" height="23" align="middle" bgcolor="#eeffff"><div
align="left">
               Username:</div></td>
             <td width="92" align="middle" bgcolor="#eeffff"><div align="right">
               <input name="UserName" type="text" value="username" size="10"
onFocus="this.value="">
              </div></td>
            </tr>
            <tr>
             <td height="11" align="middle" bgcolor="#eeffff"><div align="left">
               Password:</div>
              <div align="right"> </div></td>
             <td height="11" align="middle" bgcolor="#eeffff"><div align="right">
               <input name="Password" type="password" value="password" size="8"
onFocus="this.value="">
              </div></td>
            </tr>
            <tr>
             <td height="12" colspan="2" align="middle" bgcolor="#eeffff"><div
align="right">
```

```
                <input type="submit" name="Submit" value="Login">
              </div></td>
            </tr>
        </form>
</table>
<%
        Response.End
end if
%>
<form action="action.asp?op=rs" name="frmSell" method="post">
  <table width="600" height="237" border="0">
    <tr>
      <td height="21" colspan="2"><strong><font color="steelblue">Rent or Sale Your
        Estate</font></strong></td>
      <td height="21" colspan="2"><div align="right"></div></td>
    </tr>
    <tr>
      <td height="21" colspan="4"><hr width="100%" size="2" color="steelblue"></td>
    </tr>
    <tr>
      <td width="32%" height="26">Type of Estate</td>
      <td width="28%"><div align="right"> </div></td>
      <td><select name="frmType">
          <option value="0" selected>Select one</option>
          <option value="villa">Villa</option>
          <option value="flat">Flat</option>
          <option value="bungalow">Bungalow</option>
        </select></td>
      <td> </td>
    </tr>
    <tr>
      <td height="21">Country</td>
      <td><div align="right"> </div></td>
      <td><input type="text" name="frmCountry"></td>
```

```
  <td> </td>
</tr>
<tr>
  <td height="21">City</td>
  <td><div align="right"> </div></td>
  <td><input type="text" name="frmCity"></td>
  <td> </td>
</tr>
<tr>
  <td width="32%" height="26">Address</td>
  <td width="28%"><div align="right"> </div></td>
  <td>
            <textarea name="frmAddr" cols="20" rows="3"></textarea>
      </td>
  <td> </td>
</tr>
<tr>
  <td width="32%" height="26">Bed Rooms</td>
  <td width="28%"><div align="right"> </div></td>
  <td><select name="frmBedRoom">
     <option value="1" selected>1</option>
     <option value="2">2</option>
     <option value="3">3</option>
     <option value="4">4</option>
     <option value="5">5</option>
   </select></td>
  <td> </td>
</tr>
<tr>
  <td width="32%" height="26">Type of Estate</td>
  <td width="28%"><div align="right"> </div></td>
  <td><select name="frmBathRoom">
     <option value="1" selected>1</option>
     <option value="2">2</option>
```

```
        <option value="3">3</option>
      </select></td>
    <td> </td>
  </tr>
  <tr>
    <td height="21">How much you want? ($)</td>
    <td><div align="right"> </div></td>
    <td><input name="frmPrice" type="frmPrice" id="frmPrice" size="9"
maxlength="6"></td>
    <td> </td>
  </tr>
  <tr>
    <td height="21"> </td>
    <td><div align="right"></div></td>
    <td> <div align="left">
      <input type="radio" name="frmRentSell" value="1">
      Rent </div></td>
    <td width="17%"> <div align="left">
      <input type="radio" name="frmRentSell" value="0">
      Sale </div></td>
  </tr>
  <tr>
    <td height="21"> </td>
    <td> </td>
    <td><div align="right"> </div></td>
    <td><input type="button" name="Submit2" value="Add Estate"
onClick="DontGo()"></td>
  </tr>
 </table>

</form>
<script language="JavaScript">
        function DontGo()
                {
```

```
if (frmSell.frmAddr.value=='')
        {
                alert("Please Write Your Adress");
                return false;
        }
if (frmSell.frmCity.value=='')
        {
                alert("Please Write Your City");
                return false;
        }
if (frmSell.frmType.value=='' || frmSell.frmType.value=='0')
        {
                alert("Please Write Your Estate Type");
                return false;
        }
if (frmSell.frmCountry.value=='')
        {
                alert("Please Write Your Country");
                return false;
        }
if (frmSell.frmBedRoom.value=='')
        {
                alert("Please Write Your Bedrooms");
                return false;
        }
if (frmSell.frmBathRoom.value=='')
        {
                alert("Please Write Your Bathrooms");
                return false;
        }
if (frmSell.frmPrice.value=='')
        {
                alert("Please Write Price of Your Estate");
                return false;
```

```
                }
                else
                {
                        frmSell.submit();
                }
        }
</script>
</body>
</html>
```

REGISTRATION.ASP

```
<%@LANGUAGE="VBSCRIPT" CODEPAGE="1252"%>
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body bgcolor="#FFFFFF" onload="window.parent.status_bar.style.visibility='hidden'"
onunload="window.parent.status_bar.style.visibility='visible'"
scroll=no oncontextmenu="return false" onselectstart="return false" ondragstart="return
false">
<form name="frmReg" method="post" action="action.asp?op=ar">
 <table width="671" height="268" border="0">
  <tr>
   <td height="21" colspan="3"><font color="steelblue"><strong>Estate Agent
    Registration</strong></font></td>
   <td width="9%"> </td>
   <td width="29%"> </td>
  </tr>
  <tr>
   <td height="21" colspan="5"><hr width="100%" size="2" color="steelblue"></td>
  </tr>
  <tr>
```

```
  <td height="21" colspan="5"> </td>
</tr>
<tr align="left" valign="top">
  <td width="18%" height="21">Name Surname</td>
  <td width="22%">
    <input name="frmName" type="text" id="name" size="25"></td>
  <td width="22%"> </td>
  <td>Tel</td>
  <td>
    <input name="frmTel" type="text" id="tel"> </td>
</tr>
<tr align="left" valign="top">
  <td height="21">Company Name</td>
  <td>
    <input name="frmCoName" type="text" id="coname" size="25"></td>
  <td> </td>
  <td>Fax</td>
  <td>
    <input name="frmFax" type="text" id="fax"></td>
</tr>
<tr align="left" valign="top">
  <td height="28">City</td>
  <td>
    <input name="frmCity" type="text" id="city" size="15" maxlength="20"></td>
  <td> </td>
  <td>Mobile</td>
  <td>
    <input name="frmMobile" type="text" id="mobile"></td>
</tr>
<tr align="left" valign="top">
  <td height="26">Country</td>
  <td>
    <select name="frmCountry" id="country">
      <option value="0" selected>Select one</option>
```

```
        <option value="1">Turkey</option>
        <option value="2">Cyprus</option>
       </select></td>
     <td> </td>
     <td>E-Mail</td>
     <td>
      <input name="frmEmail" type="text" id="email"></td>
    </tr>
    <tr align="left" valign="top">
     <td height="21">Address</td>
     <td colspan="2">
      <textarea name="frmAddress" cols="25" rows="2" wrap="VIRTUAL"
id="address"></textarea></td>
     <td> </td>
     <td> </td>
    </tr>
    <tr>
     <td height="21"> </td>
     <td colspan="2"> </td>
     <td> </td>
     <td><div align="right">
       <input type="button" name="Submit" value="Submit" onclick="DontGo()">
      </div></td>
    </tr>
  </table>
</form>
<script language="JavaScript">
        function DontGo()
                {
                        if (frmReg.frmName.value==")
                                {
                                        alert("Please Write Your name");
                                        return false;
                                }
```

```
if (frmReg.frmCoName.value=='')
        {
                alert("Please Write Your coname");
                return false;
        }
if (frmReg.frmCountry.value=='' || frmReg.frmCountry.value=='0')
        {
                alert("Please Write Your country");
                return false;
        }
if (frmReg.frmFax.value=='')
        {
                alert("Please Write Your fax");
                return false;
        }
if (frmReg.frmCity.value=='')
        {
                alert("Please Write Your city");
                return false;
        }
if (frmReg.frmMobile.value=='')
        {
                alert("Please Write Your mobile");
                return false;
        }
if (frmReg.frmTel.value=='')
        {
                alert("Please Write Your tel");
                return false;
        }
if (frmReg.frmEmail.value=='')
        {
                alert("Please Write Your email");
                return false;
```

```
                }
            if (frmReg.frmAddress.value=='')
                {
                    alert("Please Write Your adress");
                    return false;
                }
            else
                {
                    frmReg.submit();
                }
        }
</script>
</body>
</html>
```

STYLE.CSS

```
BODY
{
  FONT-SIZE: 11px;
  FONT-FAMILY: arial;
  SCROLLBAR-FACE-COLOR: #ffffff;
  SCROLLBAR-HIGHLIGHT-COLOR: #ffffff;
  SCROLLBAR-SHADOW-COLOR: #a6caf0;
  SCROLLBAR-3DLIGHT-COLOR: #a6caf0;
  SCROLLBAR-ARROW-COLOR: #a6caf0;
  SCROLLBAR-TRACK-COLOR: #ffffff;
  SCROLLBAR-DARKSHADOW-COLOR: #a6caf0
}
P
{
  FONT-SIZE: 12px;
  FONT-FAMILY: arial
}
```

```
TR
{
  FONT-SIZE: 12px;
  FONT-FAMILY: arial
}
TD
{
  FONT-SIZE: 12px;
  FONT-FAMILY: arial
}
TABLE
{
  FONT-SIZE: 12px;
  FONT-FAMILY: arial
}
DIV
{
  FONT-SIZE: 12px;
  FONT-FAMILY: arial
}
INPUT
{
  BORDER-RIGHT: #4e4e4e 1px solid;
  PADDING-RIGHT: 1px;
  BORDER-TOP: #f0f0f0 1px solid;
  PADDING-LEFT: 1px;
  FONT-WEIGHT: bold;
  FONT-SIZE: 11px;
  PADDING-BOTTOM: 1px;
  TEXT-TRANSFORM: uppercase;
  BORDER-LEFT: #f0f0f0 1px solid;
  COLOR: brown;
  PADDING-TOP: 1px;
  BORDER-BOTTOM: #4e4e4e 1px solid;
```

```
   FONT-FAMILY: Tahoma, Verdana, Sans-serif;
   BACKGROUND-COLOR: #89c7f6;
   align: center
}
UNKNOWN
{
   TEXT-DECORATION: none
}
A:link
{
   COLOR: #0000ff;
   TEXT-DECORATION: none;
        font-size: 14px
}
A:visited
{
   COLOR: #ffffff;
   TEXT-DECORATION: none;
        font-size: 14px
}
A:active
{
   COLOR: #ffffff;
   TEXT-DECORATION: none;
        font-size: 14px
}
A:hover
{
   COLOR: #a00000;
   TEXT-DECORATION: underline;
        font-size: 14px
}
TEXTAREA
{
```

```
    BORDER-RIGHT: black double;
    BORDER-TOP: black double;
    FONT-SIZE: 12px;
    OVERFLOW: auto;
    TEXT-TRANSFORM: none;
    BORDER-LEFT: black double;
    COLOR: black;
    TEXT-INDENT: 10px;
    BORDER-BOTTOM: black double;
    FONT-FAMILY: 'Times New Roman';
    BACKGROUND-COLOR: #89c7f6;
    TEXT-ALIGN: justify
}
SELECT
{
    FONT-WEIGHT: bolder;
    FONT-SIZE: 10px;
    TEXT-TRANSFORM: none;
    COLOR: brown;
    BACKGROUND-COLOR: #89c7f6
}
BUTTON
{
    TEXT-TRANSFORM: none;
}
```