Active Server Pages

Objetivos

- Definir o padrão ASP;
- Compreender a arquitetura ASP;
- Definir uma plataforma mínima e ideal para o uso da tecnologia ASP;
- Introdução a programação em ASP;
- Definir e compreender a arquitetura ADO (ActiveX Data Objects)
- Acessar banco de dados com ASP;
- Utilizar Cookies com ASP;
- Compreender o uso de Objetos, Eventos e Propriedades do Active Server Pages;
- Criar uma aplicação ASP para demonstrar sua utilização.

Teoria

1. Introdução

Em Novembro de 1996, a Microsoft introduziu formalmente a tecnologia *Plataforma Ativa* no "Site Builders Conference and the Professional Developers Conference". Nesse evento, foi apresentado um diagrama que demonstra claramente os objetivos e visão da Microsoft em relação a esta tecnologia.

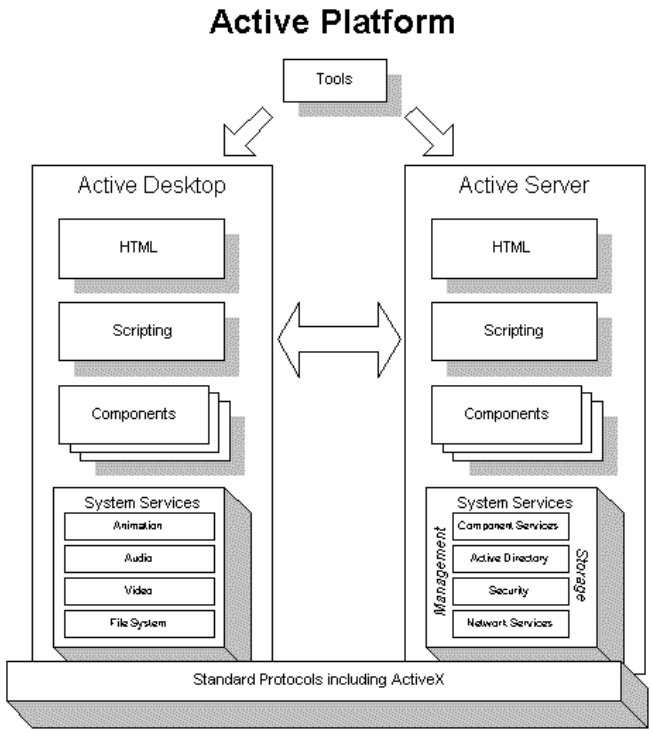


Figura 1: Visão da Microsoft sobre o Active Platform.

A Plataforma Ativa (Active Platform) é a visão da Microsoft sobre a nova geração de distribuição computacional, em relação a dados. Ela explora o melhor do modelo de programação centralizada com o melhor da programação descentralizada. A visão da Microsoft tem uma profunda implicação para a Internet e outros sistemas desenvolvidos e em desenvolvimento. O modelo de aplicações criado pela Microsoft, apresenta uma centralização no aspecto lógico e uma descentralização no aspecto físico. A centralização lógica de sistemas pode ser administrada de qualquer lugar. Quanto a descentralização do aspecto físico, é apresentada uma grande quantidade de vantagens, onde destacamos: sistemas mais eficientes, tolerância a falhas, maior poder de processamento, escalabilidade, etc.

Dois paradigmas surgem em relação a tecnologia apresentada:

1) Antes do advento do Active Server, programadores preocupavam-se com tempo de acesso e infra-estrutura a ser implementada em grades sistemas de armazenamento, além das dificuldades em estabelecer conexões confiáveis aos DBMS, exigindo um alto grau de competência por parte dos administradores e um alto grau de conhecimento de acesso a DBMS por parte dos programadores de *front-end's*;
2) Problemas na compatibilidade dos DBMS com as diversas linguagens de programação existentes no mercado, onde muitas das vezes, elas são hostis, quando deveriam ser amigáveis, visando a facilidade e eficiência no desenvolvimento.

O problema é resolvido por uma tecnologia de suporte ao desenvolvimento na Plataforma Ativa (Active Platform), o *Active Server Pages*. A tecnologia ASP (também assim chamada), é um recurso para servidores Microsoft que permite o processamento de comandos no servidor, com a conseqüente geração dinâmica de páginas HTML para o cliente.

É possível abrir bancos de dados para leitura ou alteração de registros como se fosse uma aplicação desenvolvida em Visual Basic. Usando somente ASP, podemos dispensar outras soluções como CGI, IDC/HTX, ISAPI e OLEISAPI, tornando assim mais simples o ambiente de desenvolvimento.

Com isso, resolvemos o problema da interface (front-end's) para bancos de dados, pois o browser nos fornecerá tudo o que precisamos, já que o HTML possui uma série de tag's responsáveis pela geração de formulários. Como instruções ASP não possuem uma interface, o programador não se sentirá seduzido pelos detalhes, dedicando todo o seu tempo ao centro da aplicação que é, na maior parte das vezes, o acesso ao banco de dados.

A segurança é fornecida pelo próprio servidor e/ou pelo DBMS, mas nada impede que sejam tomadas algumas precauções em relação a falhas e ao acesso, fazendo o uso de objetos internos do ASP e até de protocolos de segurança encontrados na Internet como SSL (Secure Socket Layer) e SET (Secure Eletronic Transaction).

E para finalizar, toda a tecnologia ASP está fundamentada no ActiveX, que por sua vez está fundamentada no COM/DCOM. Como linguagem, é

claro que o padrão usado é o VBScript, realizando o controle dos objetos, propriedades e eventos, tanto do lado servidor como do lado cliente.

2. ASP - Arquitetura

Quando falamos em arquitetura do Active Server Pages, devemos nos lembrar como funciona a arquitetura COM/DCOM, já que o ASP está sustentada por ela. A figura abaixo recorda bem o seu funcionamento.

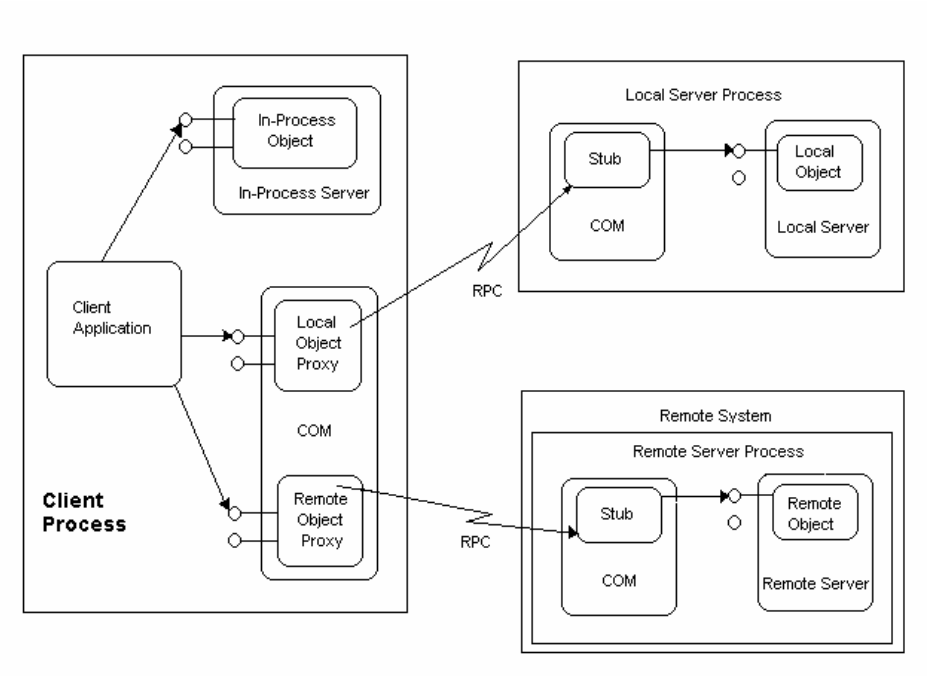


Figura 2: Modelo COM.

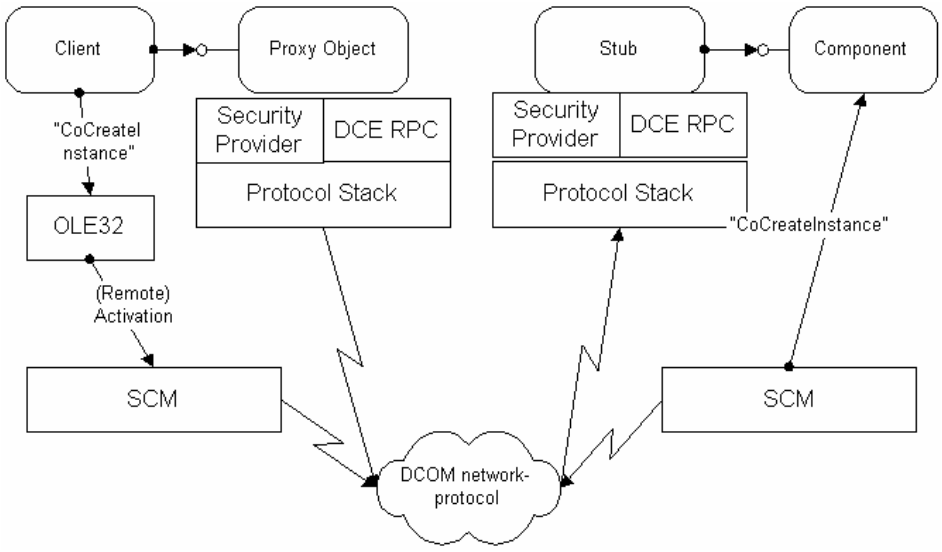


Figura 3: Modelo DCOM.

O funcionamento do ASP é bastante simples quando observamos nas camadas superiores da arquitetura Active Platform.

Quando um browser solicita uma página ASP (solicitação feita através de um link ou form), o servidor da Web sabe (se for compatível) que esta é uma página ativa, e seus processos deverão ser realizados no próprio servidor, e as respostas deverão ser enviadas em um formato legível pelo browser, ou seja, no formato HTML (tag's) e /ou script's que poderão ser executados no cliente. Os script's ASP são executados no mesmo processo do servidor por um Script Engine, sendo assim, uma vantagem para o processamento em relação a demanda de tempo.

Se a aplicação ASP realizar acesso ao banco de dados, deverá então realizar a conexão com o DBMS ou arquivos que o compõem, através de um ponteiro ODBC que deve ser criado no servidor onde está localizado o servidor Web, e mesmo deverá apontar para o local do banco de dados (o local pode ser um diretório diferente para os arquivos e/ou uma máquina diferente da que executa o servidor Web). A figura abaixo mostra o funcionamento de uma solicitação ASP realizada por um cliente.

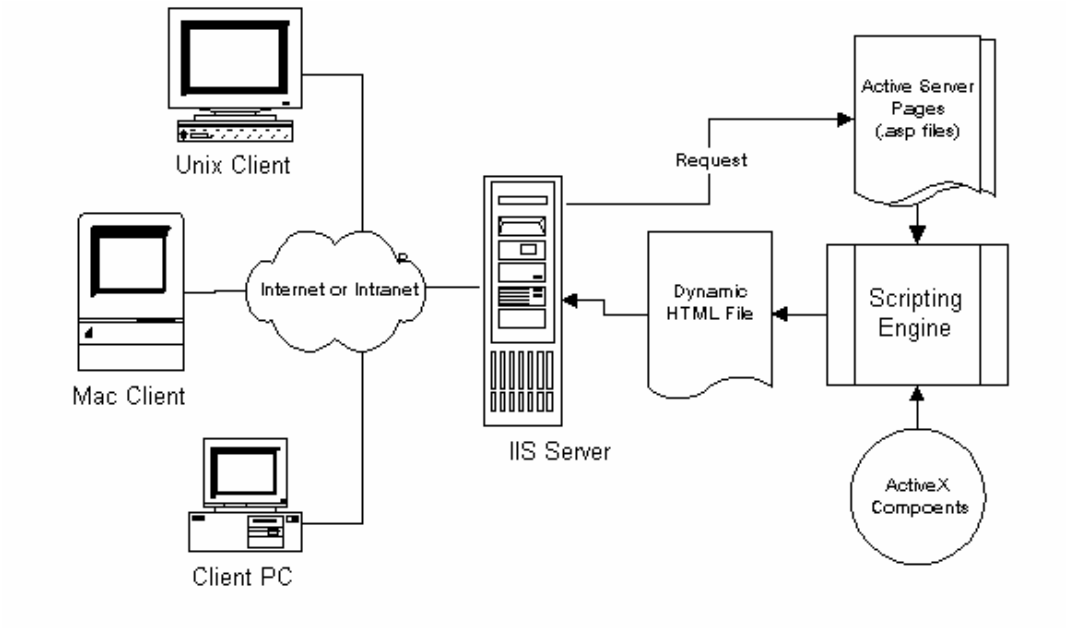


Figura 4: Funcionamento do ASP.

3. Plataforma Computacional para ASP

Do lado cliente basta um browser qualquer, desde que não se use nenhuma extensão específica, que entenda o HTML 3.2. Do lado servidor, é necessário alguns cuidados e obrigatoriamente o uso de um software servidor Web Microsoft compatível com a tecnologia ASP. O banco de dados também é fator limitante para um bom desempenho das aplicações. Tudo deverá ser dimensionado conforme a necessidade, e por isso, apresentaremos uma plataforma mínima e outra ideal para o uso da tecnologia ASP.

3.1. Plataforma Mínima

Para os desenvolvedores autônomos, é bastante difícil obter recursos computacionais e software devido ao problema de custo. Por esse motivo, apresentaremos nesse tópico um ambiente de baixo custo e de simples implementação para os testes em uma única máquina, mas com a capacidade de produzir aplicações profissionais.

O equipamento mínimo é: 486 DX2 66 MHz, 8 MB de RAM, HD com 1.2 GB e um Modem de 28.800 bps.

O software necessário é: Windows 95, com IE 3.0 instalado, Personal Web Server 1.0a, recurso ASP (ASP.EXE) instalado sobre o PWS (obtido no Option Pack para Windows NT no site da Microsoft - www.microsoft.com, Banco de Dados Microsoft Access 97 instalado, juntamente com os drives ODBC.

Um detalhe a ser observado é que o PWS 1.0a possui um bug, que ao ser instalado em um sistema em Português, o diretório criado após a instalação é o Progra~1 ao invés do Program Files. O desenvolvedor mais afoito, criará então o diretório antes da instalação (eu fiz isso) e o instalará posteriormente. Esse método provocará o aparecimento de outro bug e o sistema de segurança passará a não permitir o cadastramento de novos grupos e usuários, assim como não deixará configurar as permissões de acesso aos recursos compartilhados.

Para solucionar o problema acima, é necessário que o primeiro bug ocorra, ou seja, instale o PWS 1.0a sem a existência do diretório Program Files. Após a primeira instalação, certifique-se de que o diretório Progra~1 foi criado. Agora, antes de reinstalar o PWS, crie manualmente o diretório Program Files, e só então instale o novamente o PWS. Pronto, resolvido o problema dos bugs.

O usuário do Windows 98, não necessitará instalar o PWS 1.0a e o arquivo ASP.EXE da Microsoft, pois o Personal Web Server 4.0 (que acompanha o Windows 98) já possui todos os recursos necessários para executar ASP localmente, e detalhe, sem bugs.

Vale lembrar que não é necessário ter uma placa de rede instalada, basta instalar o recurso Acesso a Rede Dial-Up (você deve ter um modem) e os protocolos TCP/IP, e configura-los corretamente.

Para programar em ASP, não é necessário nenhum software especial, tudo pode ser feito com o Bloco de Notas e com o Access 97. Se você possuir o InterDev, será mais fácil, mas não o livrará de conhecer o padrão a fundo, pelo contrário, será necessário conhecer o padrão ASP e o ambiente de programação.

3.2. Plataforma Ideal

O hardware necessário deve ser dois computadores no mínimo, um para o servidor de banco de dados e o outro para receber o acesso dos clientes via rede local ou Internet. A configuração deve ser: Pentium II 266 MHz, 64 MB RAM, HD SCSI de 4 GB, Modem de 56 K (em apenas uma das máquinas) para a linha privativa, Placa de Rede PCI e unidade de backup, podendo ser Jaz Drive (1 GB), Fita DAT (4 GB) ou unidade de CD regravável.

O software necessário é o Windows NT 4.0 ou superior, com o Internet Information Server 3.0 ou superior, SQL Server 6.5 ou superior e browser IE 3.0 ou superior.

Esses recursos descritos acima devem ser usados em sistemas onde o fluxo de dados é constante, em situações de acesso casual ou em rajadas, o sistema pode ser o Windows NT Workstation (insisto no ambiente NT pela segurança oferecida) e o próprio Access 97, usando ODBC. Apenas tome cuidado, pois o Access 97 possui a limitação de 64 acessos simultâneos a uma banco de dados, se o seu caso o número de acessos não puder ser limitado, então faça a opção por um DBMS existente no mercado como SQL Server, DB2, Oracle, Sybase, etc.

4. Introdução a Programação em ASP

Uma aplicação ASP podemos ter diversos elementos como: Script's, ActiveX, Applets Java, Objetos internos do ASP, etc.

O ponto fundamental é termos em mente que todo o controle de rotinas e acessos a bancos de dados serão feitos por script's, para ser mais exato, um dialeto do VBScript (nada impede que nas páginas de resposta enviemos comandos JavaScript). Outro ponto a ser observado, é que em ASP, não existe uma interface como uma shell ou um form onde colocamos nossos componentes de interação com o usuário; na verdade, a interface está contida em tag's HTML e objetos Java e/ou ActiveX. Dessa forma, os comandos de controle do ASP não poderão criar se quer um msgbox/msgbox() do VB, como estamos acostumados, a resposta deverá ser dada através de uma página HTML preexistente ou *on the fly*.

4.1. Script's

Um script em ASP é identificado por um delimitador <%%>. Esse pode conter várias instruções, vejamos um pequeno exemplo:

Arquivo: Teste1.asp

Código:

```
<HTML>
<HAED>
<TITLE>Teste 1 para ASP</TITLE>
<HEAD>
<BODY>
<P> Isso é um teste de script.
<% strNome = “João” %>
<P> O nome do sujeito é <B> <% = strNome %> </B>
</BODY>
</HTML>
```

Podemos ir um pouco além e gerar mensagens mais complexas, usando fluxo de controle, veja no exemplo:

Arquivo: Teste2.asp
Código:

```
<HTML>
<HAED>
<TITLE>Teste 2 para ASP</TITLE>
<HEAD>
<BODY>
<P> Isso é um teste de script.
<%
If Time > #8:00:00AM# and Time < #5:00:00PM# Then

  strMsg= "Bem vindo ao trabalho."

Else

  strMsg = "Vá para casa, ainda não é hora de trabalhar."

End If
%>
<P> Status da hora: <% = strMsg %>
</BODY>
</HTML>
```

Nas rotinas acima, o servidor automaticamente interpreta os script’s, e envia o código HTML com o resultado automaticamente.

Se você é um bom observador e curioso, verificou que ao examinar o código fonte, a rotina ASP não aparece, mas somente a página formatada. Agora você pode esconder como é que são feitas as suas páginas (somente o código script).

Outra forma de produzir código ASP, é através de blocos de script’s. Estes estão sempre entre as tag’s <SCRIPT></SCRIPT>. Então como diferencia-los dos blocos que serão processados no cliente ?

Fácil, basta você declara-los sempre da seguinte forma:

```
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
...
</SCRIPT>
```

O detalhe, é que você é obrigado a declarar os comandos a serem executados no servidor dentro das tag's <%%> ou no formato acima, qualquer outra forma, não será interpretada no servidor, mas sim no cliente.

Outro detalhe, é que você pode diferenciar os script's de comandos isolados colocando na primeira linha do código ASP a seguinte isntrução:

<% @ LANGUAGE = **VBScript|JavaScript** %>

Mais detalhes, nunca coloque comandos isolados (aqueles <%%>) dentro de blocos de script's, pois poderá ocorrer erros de execução.

Vejamos um exemplo do uso de blocos de script's.

Arquivo: Teste3.asp
Código:

```
<% @ LANGUAGE=VBScript %>
<HTML>
<HAED>
<TITLE>Teste 2 para ASP</TITLE>
<HEAD>
<BODY>
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
        Sub Calc (x)
                response.write (x*x)
        End Sub
'Comando imediato
Calc 3
</SCRIPT>
<%
'Comando imediato
Call Calc (5)
%>
</BODY>
</HTML>
```

Mais uma vez, se você foi bom observador, viu após a execução do código acima que o comando imediato entre os delimitadores <%%>, tem prioridade em relação a execução dentro do bloco. Observou também que fora do bloco, é necessário usar os parênteses para a passagem do parâmetro.

4.2. Objetos Internos do ASP

Ao programar em ASP, os objetos do browser do cliente não estão disponíveis. Em contra partida, objetos de interação do ambiente (Objetos Internos) estão disponíveis para o uso, e são definidos como internos porque nós não os criamos, mas simplesmente os utilizamos.

Não deve ser feita nenhuma confusão entre objetos internos do ASP (que não podem ser criados ou destruídos) com objetos ActiveX do servidor (que são instâncias de objetos).

A lista abaixo apresenta os objetos internos do ASP:

Application	Representa uma aplicação ASP (Conjunto de páginas ASP de um diretório virtual do servidor);
Request	Representa os dados enviados para a página ASP por um formulário ou link do browser do cliente;
Response	Linhas da página de resposta gerada para o browser do cliente;
Server	Representação do servidor Web onde a página ASP está sendo executada. Permite o acesso a algumas propriedades e a criação de instâncias de componentes Servidores ActiveX;
Session	Representa uma sessão aberta com um cliente via browser. Quando uma sessão se fecha, todas as variáveis pertencentes a sessão encerrada são perdidas.

Todos objetos apresentados podem possuir propriedades, eventos, métodos e coleções.

4.3. Definindo uma Aplicação ASP

Uma aplicação ASP é por definição, um conjunto de arquivos ASP contido em um diretório de um servidor Web, apresentado aos usuários sob a forma de diretório virtual com permissão de execução.

A definição acima deixa uma dúvida ainda. O que une os arquivos .ASP do diretório virtual do servidor ?

Na verdade, não existe um arquivo de projeto como no Visual Basic, mas sim um arquivo (não obrigatório) chamado GLOBAL.ASA. Se este arquivo existir, então ele será executado antes de qualquer arquivo ASP.

Dentro de um arquivo GLOBAL.ASA, podemos capturar eventos de início e término de sessão e aplicação, e ainda, podemos inicializar objetos e variáveis comuns aos arquivos ASP. É importante também destacar, que somente poderá existir um arquivo GLOBAL.ASA para cada diretório virtual (aplicação).

Para iniciar uma aplicação ASP, é necessário que um usuário solicite uma página ASP através do seu browser, então ao a carregar a primeira página

ASP ou o arquivo GLOBAL.ASA a aplicação é dada como inicializada. O término da aplicação é feita somente quando o servidor Web é desligado, portanto, enquanto o servidor estiver no ar, os dados das variáveis estarão disponíveis na memória. Esse conceito é o de *Aplicação* em ASP.

Em uma aplicação executando localmente por um único usuário tem suas variáveis e objetos totalmente dedicados as suas tarefas. Mas no caso de aplicações ASP, isso nunca acontecerá, pelo contrário, poderão acessar uma mesma aplicação vários usuários simultaneamente.

Em um arquivo GLOBAL.ASA, podemos colocar *variáveis de nível de aplicação* e inicia-los com alguma informação que estará disponível enquanto o servidor Web estiver no ar.

Imaginem um cenário onde dois usuários acessem uma aplicação ASP. Se fosse necessário armazenar seus nomes, jamais poderíamos usar variáveis de nível de aplicação, pois elas seriam alteradas a cada acesso. Por exemplo:

Quando o usuário “Fulano” acessasse a variável **NomeDoUsr**, seu valor seria o nome “Fulano”. Em seguida, o usuário “Beltrano” acessa a aplicação e variável **NomeDoUsr** conteria o valor “Beltrano”. Se o usuário “Fulano” retornasse a aplicação, veria o nome “Beltrano” e não o seu nome, “Fulano.

Para resolver esse problema, em ASP, devemos usar variáveis de nível de sessão. A cada acesso, é criada uma Sessão onde o usuário possui informações exclusivas, que não estarão disponíveis para outros usuários que estejam usando a mesma aplicação. Uma *Sessão* é iniciada quando um usuário acessa uma aplicação ASP, e termina quando ele fecha o seu browser ou encerra a conexão.

Do mesmo modo que existem variáveis de *Aplicação* e *Sessão*, podemos ter Componentes Servidores ActiveX dos dois níveis. A inicialização desses componentes é feita através dos eventos **Application_OnStart**, **Application_OnEnd**, **Session_OnStart** e **Session_OnEnd**, no arquivo GLOBAL.ASA.

4.4. Objeto Application

Esse objeto nos permite manipular dados relativos a toda aplicação ASP. Esses dados podem ser compartilhados em diversas sessões por vários usuários, através de variáveis e objetos de nível de aplicação.

O objeto *Application* tem a duração e o escopo da aplicação ASP, ou seja, enquanto o servidor Web estiver no ar, o objeto estará ativo desde o primeiro acesso a aplicação.

Não possuindo propriedades, esse objeto é capaz de armazenar qualquer tipo de variável em seu interior (encapsular). Exemplo:

Sintaxe:

Application(“<NomeDaVariável”) = <Valor>

Uso:
<%Application(“NomeDoUsuario”) = “João”%>

Também podemos criar instâncias de Componentes Servidores ActiveX no interior do objeto Application. Veja a sintaxe:

```
<%
   Set Application(“<NomeDaInstânciaDeObjeto”)  =
       Server.CreateObject(“<NomeDoObjeto>”)
    Application(“<NomeDaInstânciaDeObjeto>”).NomeDoMétodo
%>
```

Em determinados casos, pode ser necessário criar instâncias locais de componentes inseridos dentro do objeto Application.

```
<%
       Set <NomeDoObjeto> =
             Application(“<NomeDaInstânciaDeObjeto>”)
       <NomeDoObjeto>.<NomeDoMétodo>
%>
```

É considerado componente, todo e qualquer objeto ActiveX registrado na máquina, e exportado por um Servidor ActiveX (antigo Servidor OLE).

O objeto Application também suporta o armazenamento de Arrays em seu interior. Exemplo:

```
<%
       Dim Vet(3)
       Vet(0) = “Papel”
       Vet(1) = “Lápis”
       Vet(2) = “Borracha”

       Application(“Material”) = Vet
%>

<%
       Estojo = Application(“Material”)
       For i = 0 to 2
             Response.write Estojo(i)
       Next
%>
```

O objeto Application possui apenas dois métodos: **LOCK** e **UNLOCK**. A função desses métodos é controlar o acesso simultâneo a variáveis e componentes armazenados dentro do objeto Application.

Como páginas ASP são executadas em ambiente compartilhado, ou seja, vários usuários acessam a mesma aplicação ao mesmo tempo, o risco do uso

simultâneo de uma mesma variável é quase 100%, podendo causar efeitos não desejáveis. Por exemplo:

Imaginem que um usuário acessa uma página com o seguinte código:

```
<%
        Application("Cont") = Application("Cont") + 1
%>
```

Se um outro usuário acessar o mesmo código ao mesmo tempo, tentando incrementar o valor da variável, o resultado seria imprevisível. O correto é refazer o código da seguinte forma:

```
<%
        Application.Lock
        Application("Cont") = Application("Cont") + 1
        Application.Unlock
%>
```

Na primeira linha o método *Lock* bloqueia o acesso por outro usuário, e após o processamento da segunda linha, o método *Unlock* desbloqueia a variável para o uso por outro usuário.

Ao acessar a primeira página de uma aplicação ASP, o evento Application_OnStart é disparado, e ao terminar a aplicação, evento Application_OnEnd é disparado.

4.5. Objeto Session

Ao se conectar a uma aplicação ASP, o usuário estará abrindo uma sessão na aplicação. Portanto, esse usuário deve contar com informações que irão lhe pertencer e serão transparentes a outros usuários.

Uma aplicação ASP é iniciada pelo evento Application_OnStart, no primeiro acesso. Logo após, o evento Session_OnStart inicia a sessão do usuário.

Durante a navegação pela aplicação ASP, a sessão do usuário estará aberta com suas variáveis locais. Se o usuário fechar o browser ou o tempo de espera se esgotar (timeout), a sessão é encerrada pelo evento Session_OnEnd.

As sessões criadas por aplicações ASP, são mantidas por elementos conhecidos por **Cookies**.

Cookies são pequenos pedaços de dados que são armazenados em um arquivo do cliente, controlado por um browser. O servidor envia esses dados (cookies) para o browser armazenar na máquina do usuário. Quando há necessidade o browser o devolve ao servidor.

Os cookies são enviados no Header HTTP para o browser do cliente. Sua sintaxe segue-se abaixo:

SET COOKIE: <nome>=<valor>; expires=<data>; path=<dirWeb>; domain=<domínio>

nome	Nome do cookie. Armazena seu valor em um nome definido pelo programador;
valor	Valor do cookie;
data	Data de expiração do cookie;
path	Diretório virtual do servidor. Sempre que o ccokie acessar alguma página deste diretório, deverá enviar este cookie de volta ao servidor;
domínio	Domínio do servidor. Sempre que o browser acessar alguma página deste domínio, deverá enviar este cookie para o servidor.

Os dados do cookie são enviados como parte do Request HTTP. Os cookies podem ser gerados e verificados pelos objetos *Request Response*.

O objeto *Session* possui duas propriedades: **SessionID** e **Timeout**. A propriedade *SessionID*, retorna o *identificador* da sessão em que o usuário Web se encontra. A propriedade *Timeout*, define o tempo máximo (default de 20 minutos) de espera por solicitações (comandos), antes de encerrar a ssessão.

<% Session.Timeout = 10 %>

O único método encontrado no objeto *Session* é o **ABANDON**. Sua função é encerrar uma sessão já estabelecida. Pode parecer redundante, já que ao encerrar (fechar) o browser, uma sessão também é encerrada. Agora, imagine um quiosque de acesso a uma intranet da empresa X, onde acessos ao seu banco de dados é intermediada por um *username* e um *password*, e o usuário ao sair não tem permissão de fechar o browser. Nesse cenário, poderia ocorrer um desastre, pois um próximo usuário mau intencionado, poderia fazer estragos no banco de dados excluindo e alterando registros sem permissão. Por motivos como esse, é necessário fornecer um mecanismo de encerramento de sessão (logout) em uma aplicação ASP.

Para ilustrar melhor os objetos Application e Session, vamos implementar um contador que funciona apenas enquanto servidor Web estiver no ar, ou seja, a cada reinicialização do servidor o contador zera. Outro detalhe muito importante, é que o contador deve contar apenas quantas pessoas visitaram o site, e não quantas visitas foram feitas ao site. O mesmo usuário não deve ser contabilizado simplesmente porque ele navegou pelo site e voltou a página inicial; e se isso acontecer o contador estará errado.

Arquivo: GLOBAL.ASA
Código:

```
<SCRIPT LANGUAGE=VBScript RUNAT=SERVER>
        Sub Application_OnStart
```

```
        Application("Hora_Inicio")
    End Sub

    Sub Session_OnStart
        Application.Lock
        If Application("x") = "" then
            Application("x") = 0
        End if

        Application("x") = Application("x") + 1
        Application.Unlock
        Session("Hora_Sessao") = time
    End Sub
</SCRIPT>
```

Arquivo: CONTADOR.ASP
Código:

```
<%@ LANGUAGE=VBScript%>
<HTML>
<HEAD>
<TITLE>Contador ASP</TITLE>
</HEAD>
<BODY BGCOLOR="WHITE">
<P><B>Hora do Início da Aplicação:</B>
<%=Application("Hora_Inicio")%>
<P><B>Hora da Sessão:</B> <%=Session("Hora_Sessao")%>
<P><B>Número de visitantes:</B> <%=Application("x")%>
</BODY>
</HTML>
```

4.6. Objeto Server

Este objeto é de suma importância na tecnologia ASP, pois ele permite a criação de instâncias de Componentes ActiveX no Servidor Web.

Sua única propriedade é *ScriptTimeout*, que determina o tempo máximo em que um script ASP poderá ficar executando no servidor. Essa propriedade é extremamente útil, pois evita que loop's inválidos ocupem o servidor permanentemente. O tempo é determinado em segundos, e o default é de 90 segundos. Exemplo:

<%Server.ScriptTimeout=20%>

Os métodos desse objeto estão listados abaixo:

CreateObject — Cria uma instância de um componente servidor ActiveX, e é somente executado no servidor Web por um Servidor ActiveX. Suas instâncias existirão enquanto a página ASP estiver sendo executada. Para destruir uma

instância, basta atribui-la a constante “Nothing”. Sintaxe:

Set Ob = Server.CreateObject(“PrgID”)

onde PrgID é o nome do Objeto como está no Registry. Exemplo:
Set V1 = Server.CreateObjetct(“MSWC.BrowserType”)

HTMLEncode Codifica uma string no formato HTML, substituindo caracteres especiais, acentos e sinais por seus equivalentes, evitando que o browser se confunda. Exemplo:

<h2><%=Server.HTMLEncode(“Ação”)%></h2>

MapPath Mapea um diretório virtual em diretório físico, de modo que possamos efetuar operações no servidor. Na verdade, esse método consulta a tabela de diretórios do servidor Web para descobrir o caminho físico. Exemplo:

sPath = Server.Path(“acessa.asp”), onde a variável sPath conteria a string, por exemplo, “c:\Web\wwwroot\acessa.asp”.

URLEncode Sua função é traduzir uma string em uma codificação URL encode, semelhante ao envio de dados por um formulário com o método GET. Exemplo:

Z = Server.URLEncode(“Teste de URL”), onde o resultado será a seguinte string: “Teste+de+URL”.

4.7. Objeto Request

Quando um browser se conecta a um servidor Web, é solicitado através de um comando interno do browser uma página HTML. Ao chamar uma aplicação ASP o mesmo ocorre, sendo assim, é enviado o *Request HTTP* para o servidor, podendo ser de duas formas: direta e indireta.

Quando a chamada é direta, simplesmente foi digitado a URL da página desejada. Na forma indireta, o usuário clicou em um link de uma página, que faz referência a uma URL diferente a local (link externo).

Independente da forma usada para chamar a página ASP, serão enviados algumas informações, e amaneira de coletar essas informações é através do

objeto Request e suas variáveis (coleções). Vejamos alguns exemplos de chamada a uma página ASP:

Formato Indireto (Link e Form)

```
<a href="http://www.microsoft.com/access.asp">Access</A>
<form action="http://www.nsw.com/log.asp" method=post>
        <input type=submit name="Login" value="Login">
</form>

<a href="livros.asp?CodLiv=2>Livro 2 – Internet</a>
```

O objeto Request não possui métodos, eventos ou propriedades apenas sua coleção de variáveis enviadas pelo usuário. A forma de acesso a essas variáveis é descrito abaixo:

```
<%
        x = Request.<coleção>("<variável>")
%>
```

Pode acontecer de não nos lembrarmos do nome da coleção a ser acessada, mas apenas do nome da variável; e nesse caso, usamos o mesmo artifício da sintaxe abaixo:

```
<%
        x = Request("<variável>")
%>
```

O ASP se encarregará de "varrer" toda a coleção do objeto para procurar pela variável com o nome especificado e retornar o seu valor.

As variáveis da coleção do objeto Request estão apresentadas abaixo:

QueryString	Valores recebidos através de um Request HTTP do tipo GET.
Form	Contém todos os campos de um formulário enviado através do método POST.
ServerVariables	Contém todas as variáveis CGI do servidor.
Cookies	Permite o acesso aos cookies do usuário Web que foram enviados. Exemplo: <% x = Request.Cookies("usuario") %> Para verificar o número de cookies existentes, podemos varrer a coleção usando a instrução **For Each ... Next**. Exemplo:

```
<% For Each CK in Request.Cookies%>
<p><%=CK%>=
<%>Request.Cookies(CK)%>
<%Next%>
```

No exemplo anterior, para cada cookie existente, seu nome será exibido seguido do seu valor, no browser do cliente.

ClienteCertificate

Permite verificar se o cliente (browser) possui certificação digital (SSL – Secure Socket Layer), assim como coletar informações a respeito da certificação. Vejamos um exemplo que verifica se o usuário possui uma certificação SSL:

```
<%
if len(Request.ClienteCertificate) = 0 then%>
<b>Cliente sem certificado</b>
<%else%>
<p>Usuário:
<%Request.ClienteCertifcate("SubjectCN")%>
<p>Nº do Certificado:
<%Request.ClienteCertifcate("SerialNumber")
%>
<%End If%>
```

As informações enviadas estão listadas a seguir:

SubjectO = Empresa do Usuário
IssuerO = Empresa Emissora do Certificado
SubjectCN = Nome do Usuário
SubjectC = País do Usuário
ValidUntil = Data de Validade do Certificado
SerialNumber = Número do Certificado

4.8. Objeto Response

A resposta a solicitação (Request HTTP) de uma página ASP, é dada pelo objeto Response. Os dados são enviados no formato HTML, por esse motivo, não é possível o usuário do browser ver como é o nosso código ASP. Para o usuário da Web, a resposta será sempre dada em HTML.

Uma característica do ASP, é que os comandos não são armazenados em um buffer de espera para posteriormente ser enviado, ou seja, tudo que não for script, o servidor enviará de imediato. Desse modo, caso deseje alterar informações antes que o servidor envie o HTML, faça-o com um script antes dos comandos HTML. O ASP até permite o uso de buffer's para comandos (Header HTTP) HTML, mas deve ser feito de forma manual.

Vejamos as propriedades do objeto Response:

Buffer — Indica se a resposta gerada deverá ser enviada automaticamente ou se deverá ser armazenada em um buffer. No caso de usar esta propriedade, o comando deverá ser usado antes de qualquer coisa na página ASP. Os valores podem ser TRUE e FALSE. Exemplo:

```
<%@ LANGUAGE=VBScript%>
<%Response.Buffer = True%>
```

No caso de usar a esta propriedade, será necessário o uso dos métodos **FLUSH** ou **END** para enviar os comandos armazenados.

ContentType — É o ContentType do formato **MIME** da página de resposta.

Expires — Usado para definir o tempo em que a página ficará no cache do usuário. No caso de páginas estáticas é bastante vantajoso que o tempo seja o mais longo possível, mas para páginas ASP, o ideal é que a atualização seja constante. Exemplo:

```
Response.Expires = 0 'Sempre atualizada
Response.Expires = 5 '5 minutos no cache
```

ExpiresAbsolute — Contém a data e hora da expiração da página. Exemplo:

```
Response.ExpiresAbsolute =
"#Jan 10, 1999 12:00:00#"
```

Status — Altera as linhas de status gerada pelo servidor Web (não é a mesma coisa da barra de status do browser). São aquelas mensagens recebidas dentro da janela do browser quando alguma coisa não funciona no servidor. Exemplo:

```
Response.Status = "500 – Erro"
```

Para o objeto Response foi definida apenas uma única coleção de dados, **Cookies**. Ele nos permitirá criar e alterar os cookies do browser dos usuários. Como essa coleção altera o Header HTTP, ele deverá ser usado antes de qualquer comando HTML em uma página ASP, a não ser que seja usada a propriedade Buffer = True. Vejamos então como criamos um cokie no browser do cliente:

<%Response.Cookies(“<variável>”) = “<valor>”%>

Podemos também acessar um atributo de um cookie e alterar o seu valor. Veja quais atributos estão disponíveis:

Expire	Data e hora de validade de um cookie.
Path	Path (caminho) do cookie na máquina do cliente.
Domain	Será enviado um cookie para todas as páginas contidas no domínio especificado.

Vejamos um exemplo da criação de um cookie e vamos também verificar se ele realmente foi criado:

```
<%
        Response.Cookie(“Logado”) = “Sim”
        Response.Cookie(“Logado”).Domain = “dwrp.eti.br”
%>
```

Para controlar e enviar informações para o browser do usuário, devemos usar os métodos do objeto Response, vejamos a lista a seguir:

AddHeader	Adiciona um Header HTTP à página gerada.
AppendToLog	Nos permite marcar informações de acesso ou segurança, adicionando um texto ao log do servidor.
BinaryWrite	Envia um dado binário para o browser. É usado quando desejamos enviar uma imagem, som, etc. Mas deverá ser informado o ContentType também. Exemplo: <% Response.ContentType = “image/gif” Figura = VerImag(“num.gif”) ‘Você deve cria essa função para ler um arquivo. Response.BinaryWrite Figura %>
Clear	Apaga o conteúdo do buffer de resposta do servidor ASP.
End	Encerra (termina) uma página ASP.
Flush	Envia o conteúdo do buffer de resposta.
Redirect	Redireciona o browser do cliente para outro endereço. Esse método altera o Header HTTP, por esse motivo deve ser usado antes de qualquer comando HTML.

Write	Permite o envio de string's para o browser do cliente. Exemplo: Resposne.Write "<H1>Teste</H1><HR>"

Vejamos um exemplo do objeto Response:

Arquivo: login.htm
Código:

```
<html>
<head>
<title>Site Privado</title>
</head>
<body bgcolor=white>
<form action = "logon.asp" method = post>
        <input type=text size=10 name=userid>
        <input type=submit value=Entrar
</form>
</body>
</html>
```

Arquivo: logon.asp
Código:

```
<%@ LANGUAGE=VBScript%>
<%
if Request.Cookies("logado") = "sim" then
        Response.Redirect "logado.htm"
End if
%>
<%
if Request.Form("userid") = "pd3" then
        Response.Cookies("logado") = "sim"
        Call Cabecalho
        Response.Write "<b> Agora você está logado</b>"
        Response.End
else
        Call Cabecalho
        Response.Write "<b> A senha é 'pd3' "
        Response.End
End if

Sub Cabecalho
        Response.Write "<html><head>"
        Response.Write "<title>Site Privado</title>"
        Response.Write "</head><body bgcolor=white>
End Sub
%>
```

Arquivo: logado.htm
Código:

```
<html>
<head>
<title>Site Privado</title>
</head>
<body bgcolor=white>
<b>Você já está logado no sistema<b>
</body>
</html>
```

5. ADO – ActiveX Data Objects

Ao analisarmos a evolução das soluções de conectividades da Microsoft para ambientes Cliente/Servidor, encontramos como base o padrão ODBC. Uma solução antecessora ao ADO foi proposta no Visual Basic 4.0 na edição Enterprise, o RDO – Remote Data Object.

De fato a solução para criação de front-end's cliente/servidor para desktop foi solucionada, porém ao se aplicar a mesma técnica sobre a Internet (Web), um verdadeiro desastre provocado por bug's ocorreram. Mesmo porque, o RDO não foi projetado com esse fim.

Uma outra solução foi a utilização da ISAPI e do OLEAPI. O primeiro prendia o programador na linguagem C++, e o segundo, apesar da possibilidade do uso do Visual Basic, era extremamente lento, pois o acesso era feito através do Jet Data Access Object (DAO – Data Access Object).

Mais recentemente, foi desenvolvida uma nova API para o acesso a banco de dados via ODBC, de forma otimizada, que foi definida como OLE DB.

A OLE DB, foi especificada com base na API do C++, criando uma interface orientada a objetos. Essa API consiste no conceito de consumidor e fornecedor de dados. Consumidores fazem uso da interface OLE DB; e fornecedores expõem seus dados através da mesma interface. Agora, o ODBC é um subsídio do OLE DB. Dessa forma, o acesso feito por OLE DB é mais rápido que DAO e RDO, com performance semelhante aos sistemas desenvolvidos em C++.

Por ser uma tecnologia baseada em ActiveX, seu uso não é restrito ao ASP, mas pode ser usado em qualquer ambiente de programação que suporte ActiveX. A figura abaixo apresenta a integração do OLE DB no ambiente de conectividade Microsoft.

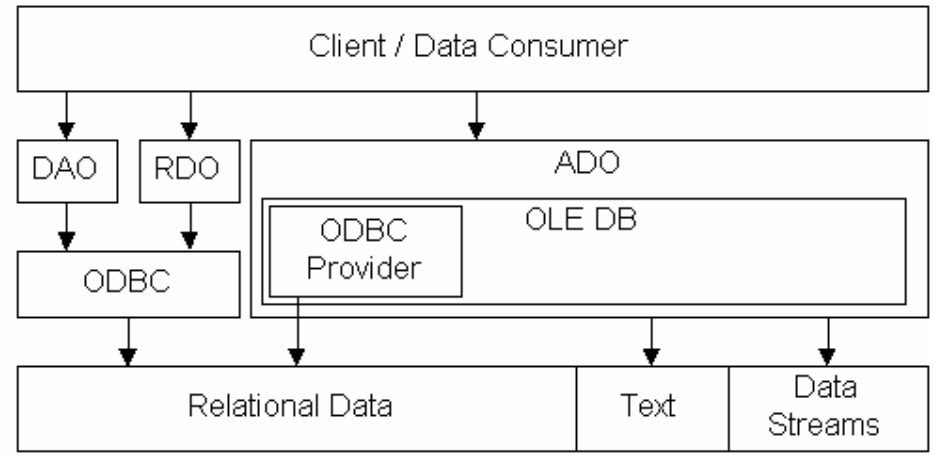
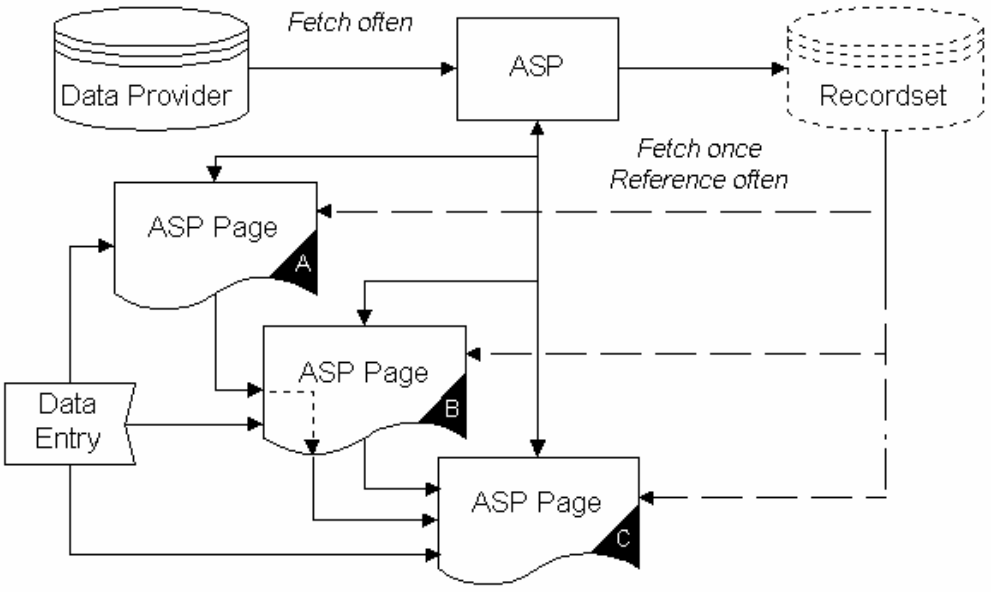


Figura 5: Integração OLE DB em soluções de conectividade Microsoft.

Ao analisarmos o modelo de programação ADO, verificamos uma grande semelhança com o velho DAO. Dessa forma pouco se altera na programação; alterando basicamente a conexão com o banco de dados e um Recordset mais eficiente. A figura abaixo apresenta o ADO sendo usado com ASP.

Figura 6: ADO e ASP.



de
car
ole
tas
ıco
lio
ım
ou
ais
eto
los

ODBC. A execução desse objeto é feita através de uma Connection, onde a resposta gerada poderá ser um Recordset. Os comandos devem ser escritos em linguagem SQL compatível com o padrão ODBC. Vejamos um exemplo de instância desse objeto:

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

	Inst.CommandText = “SELECT * FROM Clientes WHERE Devedor = ‘Sim’ ” Set Inst.ActiveConnection = DB Set RS = DB.Execute
Recordset	Objeto que representa um conjunto de registros resultantes do processamento de uma instrução SQL. Existe 3 formas de gerar um Recordset: **Método Execute de uma Connection:** Set RS = DB.Execute(“SELECT * FROM Cli”) **Método Execute de um Command:** Inst.CommandText = “SELECT * FROM Cli” Set RS = Inst.Execute **Método Open do Recordset:** Set RS = Server.CreateObjetc(“ADODB.Recordset”) RS.Open Inst
Field	Esse objeto está associado à Coleção Fields, representando os campos de um Recordset. Exemplo: Set RS = DB.Execute(“SELECT * FROM Cli”) X = RS(“Nome”) Y = RS(“Telefone”) RS(“Nome”) = “Fulano”

5.1. Objeto Connection

Para criarmos um objeto Connection, devemos ter um datasource ODBC criado previamente. O exemplo abaixo demonstra a criação e associação do objeto com o datasource ODBC:

```
Set wm = Server.CreateObject("ADODB.Connection")
DataSource = "webmail"
wm.Open DataSource, "admin", ""
```

No caso de usarmos um sistema como o Windows NT e o SQL Server, poderíamos também definirmos os parâmetros ODBC manualmente, dispensando a criação de datasource. Porém, é recomendável que seja criado um datasource sempre, evitando alterações de parâmetros dentro dos script’s ASP. Veja o exemplo de definição de parâmetros manualmente:

```
Wm.Open “DRIVER={SQL Server];SERVER=dwrp;” &_
“UID=admin;PWD=;DATABASE=webmail”
```

Não é necessário explicitar uma instância do objeto Connection, pois tanto o objeto Command como o Recordset, possuem um parâmetro chamado **ActiveConnection** que permite a criação implícita de um objeto Connection.

Vejamos agora as principais propriedades do Objeto Connection:

CommandTimeout — Tempo máximo de espera pela execução de um comando. O Default é 30 segundos.

ConnectionString — String de conexão ao banco de dados ODBC. Pode ser o nome do datasource ou os parâmetros ODBC. Essa propriedade pode ser omitida caso seja usado o método Open, informando diretamente nele o ConnectionString.

Connection.Timeout — Tempo máximo para tentar abrir uma conexão. O default é de 15 segundos.

Agora, vamos ver os principais métodos:

Open — Abre uma Connection iniciando uma tentativa de conexão com o Banco de Dados. Sintaxe:

```
<conexão>.Open
        <ConnectionString>,
        <usuário>,
        <password>
```

Close — Fecha uma conexão ativa, liberando assim os recursos no servidor de Banco de Dados. Sintaxe:

```
<Connection>.Close
```

Execute — Esse método executa um comando através de um objeto Connection. Sintaxe:

```
<Connection>.Execute
        "String SQL|Stored Procedure"
```

BeginTrans — Inicia uma transação no Banco de Dados, caso seja possível. A vantagem de utilizarmos transações, é que as operações são executadas em um buffer, e somente quando a transação é concluída é que o buffer é gravado no

Banco de Dados. Se ocorrer algum erro as operações contidas no buffer são descartadas imediatamente, evitando a gravação de dados órfãos. Vejamos um exemplo:

```
'Insere o novo usuário na tabela
On Error Resume Next
Comando = "INSERT INTO Usuarios (Nome, Curso, Turma, NomeDoUsuario, Senha)" & "VALUES('" & Nome & "','" & Curso & "','" & Turma & "','" & NomeDoUsuario & "','" & Senha & "')"
wm.BeginTrans
wm.Execute(Comando)
if Err then
        wmRollBackTrans
else
        wm.CommitTrans
end if
```

CommitTrans — Encerra uma transação com sucesso, efetuando no Banco de Dados todas as operações armazenadas no buffer.

RollBackTrans — Finaliza uma transação ignorando todas operações do BeginTrans.

5.2. Objeto Recordset

Objeto responsável pelo acesso aos campos e dados de uma ou mais tabelas em um Banco de Dados. Vejamos então suas propriedades:

ActiveConnection — Aponta para o objeto Connection através do qual a instância do objeto Recordset foi ou será gerado. Exemplo:

```
'Conexão com o Banco de Dados
Set wm = Server.CreateObject("ADODB.Connection")
DataSource = "webmail"
wm.Open DataSource, "admin", ""

'Procura pela mensagem
Comando = "SELECT * FROM Mensagens WHERE IDmsg = " & Cdbl(IDmsg)
Set RecSet = wm.Execute(Comando)
if not RecSet.Eof then
```

```
        Comando = "UPDATE
        Mensagens SET Lido = True
        WHERE IDmsg = " &
        Cdbl(IDmsg)
        wm.Execute(Comando)
End if
```

No exemplo acima, a ActiveConnection é o próprio **wm**. Poderíamos ter criado o objeto de outra forma:

```
RecSet.ActiveConnection = “webmail”
```

Ou

```
Set RecSet =
Server.CreateObject(“ADODB.Recordse
t”)
Set RecSet.ActiveConnection = wm
RecSet.Open = "SELECT * FROM
Mensagens WHERE IDmsg = " &
Cdbl(IDmsg)
```

No primeiro caso, o Recordset foi criado diretamente com base no datasource ODBC, e no segundo caso, foi indicado um objeto Connection já existente.

CursorType

Define um tipo de ponteiro para um Banco de Dados. Ao criarmos um Recordset, estamos efetivamente abrindo um Cursor (ponteiro) no Banco de Dados.

Se lembrarmos da tecnologia DAO, utilizada no Visual Basic, temos as opções de cursores do tipo: dbOpenTable, dbOpenDynaset e dbOpenSnapshot. No ADO, temos os seguintes tipos de cursores:

ForwardOnly – Tipo 0:
Ao utilizar este tipo, será gerada uma cópia física e estática dos registros selecionados, em sua estação de trabalho. A vantagem é a performance na movimentação entre os dados, porém somente movimentações poderão ser realizadas, ou seja, nada que altere ou adicione dados. Outra desvantagem, é que ao chegar ao final

do Recordset, não poderemos mais voltar aos registros anteriores.

Keyset – Tipo 1:
Esse tipo de cursor permite o acesso imediato as alterações realizadas por outros usuários, porém não será possível ver as inclusões feitas por outros usuários. Pode-se navegar para frente ou para trás nesse tipo de Recordset.

Dynamic – Tipo 2:
Permite enxergar qualquer alteração ou inclusão realizada por outros usuários, também possibilitando a navegação em todas as direções.

Static – Tipo 3:
Semelhante ao ForwardOnly, porém permite a navegação em todas as direções.

A determinação do tipo de cursor utilizado pelo Recordset é feita antes de usar o método Open.

BOF e EOF

Indica se o foi alcançado o início ou o fim do Recordset, respectivamente. Nos dois casos, se o valor retornado for TRUE indica o estado. Exemplo:

```
<%if Not RecSet.Eof then%>

<form>
<input language="JavaScript" type="button" value="Responder" onclick="Window.location.href = 'responde.asp?IDmsg=<%=IDmsg%>'" name="Responder">
<input language="JavaScript" type="button" value="Excluir Mensagem" onclick="Window.location.href = 'apaga.asp?IDmsg=<%=IDmsg%>'" name="Exclui">
<input language="JavaScript" type="button" value="Voltar para Mensagens" onclick="Window.location.href = 'mensagem.asp'" name="Voltar">
```

```
</form>
<TABLE BORDER=0>
<TR>
<TD><B>De:</B></TD><TD><%=Re
cSet("De")%></TD>
</TR>
<TR>
<TD><B>Para:</B></TD><TD><%=
RecSet("Para")%></TD>
</TR>
<TR>
<TD><B>Assunto:</B></TD><TD><
%=RecSet("Assunto")%></TD>
</TR>
<TR>
<TD
COLSPAN=2><B>Conteúdo:</B></T
D><TR><TD
COLSPAN=2><%=RecSet("Conteudo")
%></TD></TR>
</TR>
</TABLE>
<%Else%>
<H3><B>Erro:</B> Não foi
encontrada a mensagem!</H3>
<%End if%>
```

Filter

Filtra o conteúdo de um Recordset, permitindo a navegação somente entre os registros que satisfaçam as condições do filtro. Exemplo:

```
RecSet.Filter =
“IDmsg = ” & Cdbl(IDmsg)
```

Para desfazer o filtro basta atribuir a esta propriedade o valor “”. Exemplo:

```
RecSet.Filter = “”
```

LockType

É possível que ao acessarmos um determinado registro em um Banco de Dados, outros usuários o esteja fazendo ao mesmo tempo. Essa propriedade permite a administração desse problema:

ReadOnly – Tipo 1:
Acesso somente para leitura.

Pessimistic – Tipo 2:
Atualização com travamento a cada registro, no momento da leitura.

Optimistic – Tipo 3:
Autalização com travamento apenas no momento de atualizar, com o método Update.

Os métodos que serão apresentados a seguir, podem não ser suportados por todos SGBD's, mas com certeza o MS – Access aceitará todos eles. Para verificar se um determinado método é suportado ou não, utilize o método **Supports**. Utilize preferencialmente comandos SQL para adicionar ou atualizar registros.

AddNew — Adiciona um registro em branco ao Recordset.

Close — Fecha um Recordset.

Delete — Elimina um recordset.

MoveFirst, MoveLast, MoveNext e MovePrevious — Métodos usados para a navegação entre os registros de um Recordset. Vale lembrar que ao abrirmos um Recordset, o ponteiro de leitura do Banco de Dados já estará apontando para o primeiro registro da tabela.

Open — Abre um Recordset acessando uma tabela do Banco de Dados. Sua sintaxe:

```
<Recordset>.Open <SQL>,
                 [<Connection>],
                 [<CursorType>],
                 [<Lock>]
```

Vejamos o significado dos parâmetros:

SQL:
Pode ser um objeto Command ou uma instrução SQL;

Connection:
A Connection a ser usada para abrir o Recordset.

	CursorType: Tipo de cursor desejado. Deve ser passado o número e não o nome do tipo. **Lock:** Define o tipo de Lock (travamento) a ser usado nas operações com o Recordset. Deve ser passado o número e não o nome do tipo.
Update	Método utilizado para alterar o Recordset corrente.
Supports	Verifica se um determinado método é suportado pelo SGBD. Sintaxe: <Recordset>.Supports(<código do método>) O Retorno será um valor TRUE ou FALSE. O código dos métodos estão listados a seguir: AddNew – **16778240** Delete – **16779264** MovePrevious – **512** Update – **16809984**

5.3. Objeto Command

A função desse objeto é permitir a criação e execução de comandos em um Banco de Dados. Vejamos as suas principais propriedades:

CommandText	Contém o texto SQL do comando a se executado.
CommandTimeout	Tempo máximo de espera pela execução de um comando. Default de 30 segundos.
ActiveConnection	Análogo a propriedade do objeto Recordset.

O método mais importante desse objeto é o **Execute**. Veja sua sintaxe:

<comando>.Execute

Ou

Set <Recordset> = <comando>.Execute

5.4. Objeto Field

Esse objeto tem como função representar um campo de um Recordset. Suas principais propriedades são:

ActualSize — Tamanho do conteúdo atual do campo.

DefinedSize — Tamanho definido para o campo.

Name — Nome do campo.

Type — Tipo de dados suportado pelo campo, segundo o ADO. Os principais tipos são:

Small – Tipo 2:
Inteiro, 2 bytes.

Integer – Tipo 3:
Inteiro, 4 bytes.

Long - Tipo 20:
Inteiro, 8 bytes.

Boolean – Tipo 11:
True/False.

String – Tipo 129:
String.

Currency – Tipo 6:
Moeda.

Date – Tipo 7:
Tipo Data.

DBDate – Tipo 133:
Data no formato “aaaammdd”.

Time – Tipo 134:
Hora no formato “hhmmss”.

Value — Valor ou conteúdo do campo. Essa é propriedade default do objeto, podendo ser omitida em operações de atribuição de valor ao campo ou retorno de informações contidas no campo. Exemplo:

```
A = RecSet(“Nome”)
RecSet(“Nome”) = “João"
RecSet(“Nome”).Value = “Silvia”
```

Esse objeto possui apenas dois métodos: **AppendChunk e GetChunk**. O objetivo é permitir a manipulação de campos longos em formato binário.

6. Aplicação Prática dos Recursos ASP

Uma das melhores maneiras de treinar o que se aprende em uma linguagem de programação é desenvolver uma aplicação que envolva todos os recursos aprendidos, ou pelo menos uma grande parte deles.

Nesse tópico construiremos um serviço de WebMail. Seu objetivo é permitir que um usuário crie sua conta e use os principais recursos de um serviço de mensagem (Enviar, le, responder, listar e eliminar). Para esta aplicação, não serão usados os protocolos SMTP e POP3, os dados estarão armazenados em um banco de dados MS - Access 97.

Para facilitar o nosso trabalho vamos dividir em etapas o desenvolvimento da nossa aplicação.

1ª Parte: Criação do Banco de Dados

Primeiramente vamos abrir o Access e o nosso Banco de Dados conforme a documentação abaixo:

Tabela: Mensagens

Nome	**Tipo**	**Tamanho**
IDmsg	AutoNumeração	
Atributos:	Tamanho fixo, AutoIncrementar, Chave-Primária	
NomeDoUsuario	Texto	50
Atributos:	Comprimento variável	
DataDaMsg	Data/Hora	8
Formato:	dd/mm/yyyy	
Máscara de entrada:	##/##/####	
De	Texto	50
Para	Texto	50
Assunto	Texto	50
Conteudo	Memorando	-
Lido	Sim/Não	-

Tabela: Usuarios

Nome	**Tipo**	**Tamanho**
Nome	Texto	50
Curso	Texto	30
Turma	Texto	10
NomeDoUsuario	Texto	50
Senha	Texto	10

Relacionamentos

Tabela			**Tabela**
Usuarios	*Relacionamento*		*Mensagens*
Campo	**Tipo**	**Tipo**	**Campo**
NomeDoUsuario	1	Muitos	NomeDoUsuario

Atributos: Vigente, Atualizações em cascata, Exclusões em cascata

Atributos: Um-para-muitos

Permissões de usuário
admin
Permissões de grupo
Admins
Users

O banco de dados deverá ter um datasource chamado “webmail”. Para isso, vá ao Painel de Controle, ODBC de 32bits, selecione a guia NFD do Sistema, clique no botão “Adicionar”, na caixa que apareceu defina os parâmetros conforme a figura abaixo (o path do Banco de Dados na sua máquina pode ser diferente, por isso, informe a localização conforme encontra-se no seu computador). Clique “OK” nas demais janelas e feche o ODBC.

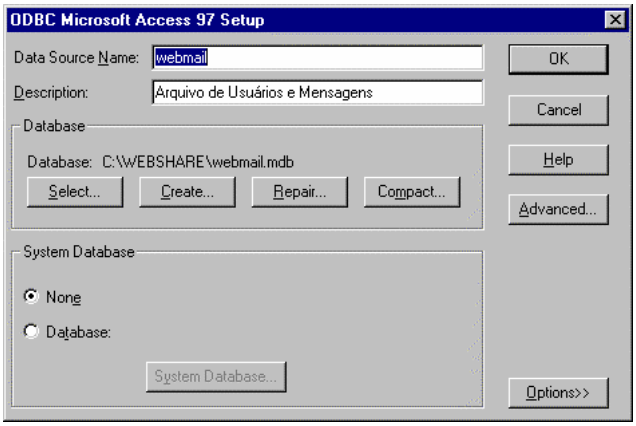


Figura 7: Criação de um datasource.

2ª Parte: Criação das páginas estáticas do site (HTML puro)

Agora iremos criar as páginas que chamam os script's ASP e as páginas que são chamadas por eles.

Arquivo: index.htm

Descrição:

Essa página é a Home Page do site WebMail, ela define as operações iniciais em nossa aplicação ASP.

Código:

```
<html>
<head>
<title>WebMail ORT</title>
</head>
<body background="fundo_.gif">
<p style="background-color: rgb(255,255,0)"><font
color="#000080"><big><big><strong>Instituto
de Tecnologia ORT (Intranet)</strong></big></big></font></p>
<p align="right"><font
color="#800000"><big><em><big><big><strong>WebMail
ORT</strong></big></big></em></big></font></p>
<hr>
<table border=0>
  <form action="acessa.asp" method="POST">

    <tr><td>User Name:</td>
    <td><!--webbot bot="Validation" b-value-required="TRUE" i-
minimum-length="4"
    i-maximum-length="50" --><input type="text" size="20"
name="NomeDoUsuario" maxlength="50"></td>
    <td>Senha:</td>
    <td><!--webbot bot="Validation" b-value-required="TRUE" i-
minimum-length="4"
    i-maximum-length="10" --><input type="password" size="20"
name="Senha" maxlength="10"></td>
    <td><input type="submit" value="Logar"
name="B1"></td><td><input type="reset" value="Limpar"
    name="B2"></td><td><input language="JavaScript" type="button"
        value="Nova Conta" onclick="Window.location.href =
'cadastra.htm';" name="B3"></td></tr>
  </form>
</table>
<hr>
<h2>Bem Vindo !</h2>
```

```
<p align=left>Esse é o mais novo serviço do laboratório de informática do
ORT, desenvolvido para você aluno, professor ou funcionário.</p>
<p align=left>O nosso WebMail é bastante simples de usar, crie uma
nova conta para você clicando no botão acima "Nova Conta" e preencha o
cadastro corretamente. Para logar no sistema, basta informar o seu User
Name (que é você quem escolhe na hora de se cadastrar) e sua
senha.</p>
<p align=left>Para que esse serviço esteja sempre disponível para você,
use-o com consciência, evitando brincadeiras e enchendo a caixa postal
dos usuários com bobagens. O serviço é público, e não monitoramos suas
mensagens. Mas a cada 30 dias limpamos todas as caixas postais, e uma
forma de não perder uma mensagem importante é salvando a página com
a mensagem em sua conta na rede do ORT.</p>
<p align=left><b>Administrador</b>
<hr>
<p><small><font color="#000080">Copyright © 1998 Daniel Wander
Ribeiro Pereira<br>
Instituto de Tecnologia ORT<br>
Última Atualização: 15/10/1998</font></small></p>
</body>
</html>
```

Comentários:

Nesse HTML observamos a existência de um formulário que chama a página ASP “acessa.asp” no parâmetro ACTION. Encontramos um terceiro botão, com uma instrução JavaScript, chamando a página “cadastra.htm”.

Arquivo: cadastra.htm

Descrição:

Essa página é a responsável pelo cadastro de novas contas no serviço de WebMail.

Código:

```
<html>
<head>
<title>WebMail ORT: Abertura de Conta</title>
</head>
<body background="fundo_.gif">
<p style="background-color: rgb(255,255,0)"><font
color="#000080"><big><big><strong>Instituto
de Tecnologia ORT (Intranet)</strong></big></big></font></p>
<p align="right"><font
color="#800000"><big><em><big><big><strong>WebMail
ORT</strong></big></big></em></big></font></p>
<hr>
```

```
<p><font color="#000080"><big><em><big><big>Abertura de Nova
Conta</big></big></em></big></font></p>
<form action="cadastra.asp" method="POST">
<table border = 0>
<tr>
        <td><input type="submit" value="Cadastrar" name="B1"></td>
        <td><input type="reset" value="Cancelar" name="B2"></td>
        <td><input language="JavaScript" type="button" value="Voltar"
onclick="Window.location.href = 'index.htm'" name="Voltar"></td>
</tr>
<tr>
        <td> </td>
</tr>
<tr>
        <td>Nome completo:</td>
        <td><input type="text" size="20" name="Nome"></td>
</tr>
<tr>
        <td>Curso:</td>
        <td><select size="1" name="Curso">
                <option value="BT"> Processamento de Dados </option>
                <option value="EL"> Eletrônica </option>
                <option value="PD"> Biotecnologia </option>
                <option value="ORT"> ORT </option>
        </select></td>
</tr>
<tr>
        <td>Turma:</td>
        <td><select size="1" name="Turma">
                <option value="PD1A">PD1A</option>
                <option value="PD1B">PD1B</option>
                <option value="PD2A">PD2A</option>
                <option value="PD2B">PD2B</option>
                <option value="PD3A">PD3A</option>
                <option value="PD3B">PD3B</option>
                <option value="BT1">BT1</option>
                <option value="BT2">BT2</option>
                <option value="BT3">BT3</option>
                <option value="EL1">EL1</option>
                <option value="EL2">EL2</option>
                <option value="EL3">EL3</option>
                <option value="Professores">Professores</option>
                <option value="Funcionários">Funcionários</option>
        </select></td>
</tr>
<tr>
        <td>User Name:</td>
        <td><input type="text" size="20"
name="NomeDoUsuario"></td>
</tr>
<tr>
```

```
        <td>Senha:</td>
        <td><!--webbot bot="Validation" startspan b-value-
required="TRUE" i-minimum-length="4"
    i-maximum-length="10" --><!--webbot bot="Validation" endspan --
    ><input type="password"
        size="20" name="Senha" maxlength="10"></td>
    </tr>
    </table>
    </form>
    <hr>
    <p><small><font color="#000080">Copyright © 1998 Daniel Wander
    Ribeiro Pereira<br>
    Instituto de Tecnologia ORT<br>
    Última Atualização: 15/10/1998</font></small></p>
    </body>
    </html>
```

Comentários:

Observando o código acima, encontramos um formulário responsável por chamar o script ASP “cadastra.asp”. Vemos novamente um terceiro botão, com uma isntrução JavaScript, chamando a página inicial, “index.htm”.

Arquivo: negado.htm

Descrição:

Essa página é a apresentada ao usuário toda vez que ele tenta logar no site com um nome ou senha inválida.

Código:

```
    <html>
    <head>
    <title>WebMail ORT: Acesso Negado</title>
    </head>
    <body background="fundo_.gif">
    <p style="background-color: rgb(255,255,0)"><font
    color="#000080"><big><big><strong>Instituto
    de Tecnologia ORT (Intranet)</strong></big></big></font></p>
    <p align="right"><font
    color="#800000"><big><em><big><big><strong>WebMail
    ORT</strong></big></big></em></big></font></p>
    <hr>
    <center>
    <table border=3>
        <tr>
        <td><h2><font color="#FF0000">Acesso Negado
!</font></h2></td>
        </tr>
```

```
</table>
</center>
<form>
  <div align="center"><center><p><input language="JavaScript"
type="button" value="Voltar"
  onclick="Window.location.href = 'index.htm'" name="Voltar"> </p>
  </center></div>
</form>
<hr>
<p><small><font color="#000080">Copyright © 1998 Daniel Wander
Ribeiro Pereira<br>
Instituto de Tecnologia ORT<br>
Última Atualização: 15/10/1998</font></small></p>
</body>
</html>
```

Comentários:

O único detalhe, e um formulário que apresenta apenas um botão, com uma instrução JavaScript, apontando para a página inicial, “index.htm”.

3ª Parte: Criação das páginas ASP

Uma vez que já temos o Banco de Dados e as páginas estáticas, já podemos dar início a construção das páginas ASP.

Arquivo: cadastra.asp

Descrição:

Esse script tem a função de cadastrar um novo usuário no serviço de WebMail.

Código:

```
<%@ LANGUAGE = VBScript %>
<html>
<head>
<title>WebMail ORT: Cadastra</title>
</head>
<body background="fundo_.gif">
<p style="background-color: rgb(255,255,0)"><font
color="#000080"><big><big><strong>Instituto
de Tecnologia ORT (Intranet)</strong></big></big></font></p>
<p align="right"><font
color="#800000"><big><em><big><big><strong>WebMail
ORT</strong></big></big></em></big></font></p>

<hr>
```

```
<%
'Declaração e inicialização de variáveis
Dim Nome, Curso, Turma, NomeDoUsuario, Senha, NewUser

Nome = request.form("Nome")
Curso = request.form("Curso")
Turma = request.form("Turma")
NomeDoUsuario = request.form("NomeDoUsuario")
Senha = request.form("Senha")

NewUser = NomeDoUsuario

'Conexão com o Banco de Dados
Set wm = Server.CreateObject("ADODB.Connection")
DataSource = "webmail"
wm.Open DataSource, "admin", ""

'Insere o novo usuário na tabela
Comando = "INSERT INTO Usuarios (Nome, Curso, Turma,
NomeDoUsuario, Senha)" & "VALUES('" & Nome & "','" & Curso & "','" &
Turma & "','" & NomeDoUsuario & "','" & Senha & "')"
wm.Execute(Comando)

'Lendo o registro recém-inserido
Comando = "SELECT * FROM Usuarios WHERE NomeDoUsuario LIKE
'" & trim(NomeDoUsuario) & "'"
Set RecSet = wm.Execute(Comando)
%>

<H3>User Name: <I><%= RecSet("NomeDoUsuario")%></I>
Inserido!</H3>
<P> Sua senha de acesso é <B><%= RecSet("Senha")%></B>,
use-a para logar no WebMail do ORT.
<P>
<form>
<!-- Envia uma nova mensagem de boas vindas para o novo usuário e
retorna a página index.htm -->
<div
  align="center"><center><p><input type="submit" value="Voltar"
language="JavaScript" onClick="Window.location.href =
'msgnew.asp?NewUser=<%=NewUser%>'" name="Voltar"> </p>
  </center></div>
</form>

<hr>

<p><small><font color="#000080">Copyright © 1998 Daniel Wander
Ribeiro Pereira<br>
Instituto de Tecnologia ORT<br>
Última Atualização: 15/10/1998</font></small></p>
```

```
</body>

</html>
```

Comentários:

Por motivos de formatação, algumas linhas encontram-se separadas. Na hora que você estiver digitando, não sepere-as.

É interessante ressaltar nesse script, que após o novo usuário ser cadastrado no sistema, automaticamente, após o usuário clicar no botão “Voltar”, para retornar a página inicial (index.htm), um script ASP é disparado para enviar uma mensagem de boas vindas ao novo usuário.

Arquivo: msgnew.asp

Descrição:

Sua função é enviar uma mensagem de boas vindas ao novo usuário do serviço de WebMail.

Código:

```
<%@ LANGUAGE = VBScript %>

<%
'Declaração e inicialização de variáveis
Dim NomeDoUsuario, DataDaMsg, De, Para, Assunto, Conteudo

NomeDoUsuario = request("NewUser")
DataDaMsg = Date()
De = "WebMail"
Para = NomeDoUsuario
Assunto = "Bem Vindo"
Conteudo = "<p><font color=blue>Bem vindo ao nosso sistema de
WebMail</font>. <p>Esperamos que goste do nosso serviço. <p>Após
ler essa mensagem apague-a. Obrigado."

'Conexão com o Banco de Dados
Set wm = Server.CreateObject("ADODB.Connection")
DataSource = "webmail"
wm.Open DataSource, "admin", ""

'Procura pelo destinatário
Comando = "SELECT * FROM Usuarios WHERE NomeDoUsuario = '" &
trim(Para) & "'"
Set RecSet = wm.Execute(Comando)

if Not RecSet.Eof then
        'Insere a nova mensagem
```

```
        Comando = "INSERT INTO Mensagens (NomeDoUsuario,
DataDaMsg, De, Para, Assunto, Conteudo)" & "VALUES('" & Para & "','"
& DataDaMsg & "','" & De & "','" & Para & "','" & Assunto & "','" &
Conteudo & "')"
        wm.Execute(Comando)
End if
response.redirect "index.htm"
%>
```

Comentários:

Nesse script, vemos que não há uma montagem de uma página HTML para o usuário após a sua execução. Simplesmente, ao término da execução é apresentada a página “index.htm”.

Outra curiosidade, é que a mensagem enviada para o novo usuário, recebe formatações HTML. Isso mesmo, ao utilizar o serviço, o usuário poderá formatar suas mensagens com tag’s HTML.

Arquivo: acessa.asp

Descrição:

A função dessa página ASP é realizar a validação do usuário no sistema para que ele tenha acesso a suas mensagens e ao serviço.

Código:

```
<%@ LANGUAGE = VBScript %>

<%
'Declaração e inicialização de variáveis
Dim NomeDoUsuario, Senha

NomeDoUsuario = request.form("NomeDoUsuario")
Senha = request.form("Senha")

'Conexão com o Banco de Dados
Set wm = Server.CreateObject("ADODB.Connection")
DataSource = "webmail"
wm.Open DataSource, "admin", ""

'Obtém o registro do usuário
Comando = "SELECT * FROM Usuarios WHERE NomeDoUsuario = '" &
trim(NomeDoUsuario) & "' AND Senha = '" & trim(Senha) & "'"
Set RecSet = wm.Execute(Comando)

if RecSet.Eof then
        response.redirect "negado.htm"
else
        Session("NomeDoUsuario") = NomeDoUsuario
```

```
        Session("Senha") = Senha
        response.redirect "mensagem.asp"
End if
%>
```

Comentários:

Caso o usuário forneça um nome e/ou uma senha inválida, eele será redirecionado para a página “negado.htm”, informado que ele não tem acesso ao sistema.

Outra observação, é o fato da página redirecionar o usuário validado para outro script sem apresentar interface (código HTML).

Arquivo: mensagem.asp

Descrição:

Essa é a principal página ASP do sistema. Ela apresenta as mensagens recebidas e seu status (lida ou não lida). Permite também, enviar uma nova mensagem para outro usuário do sistema.

Código:

```
<%@ LANGUAGE=VBScript %>
<%'Proteção para impedir acesso direto:
if Session("Senha") = "" then
        response.redirect "index.htm"
End if
%>
<html>

<head>
<meta name="GENERATOR" content="Microsoft FrontPage 3.0">
<title>WebMail ORT: Suas Mensagens</title>
</head>

<body background="fundo_.gif">

<p style="background-color: rgb(255,255,0)"><font
color="#000080"><big><big><strong>Instituto
de Tecnologia ORT (Intranet)</strong></big></big></font></p>

<p align="right"><font
color="#800000"><big><em><big><big><strong>WebMail
ORT</strong></big></big></em></big></font></p>

<hr>
```

```
<%
'Conexão para o Banco de Dados
Set wm = Server.CreateObject("ADODB.Connection")
DataSource = "webmail"
wm.Open DataSource, "admin", ""

'Criando um recordset para ler os registro do usuário
Comando = "SELECT * FROM Mensagens WHERE NomeDoUsuario = '"
& trim(Session("NomeDoUsuario")) & "'"
Set RecSet = wm.Execute(Comando)
%>

<TABLE BORDER=0>
<TR>
<TD>
<form>
<p><input type="button" value="Nova Mensagem" name="B1"
LANGUAGE="JavaScript" onclick="Window.location.href =
'nova.asp'"></p>
</form>
</TD>
<TD>
<form>
<p><input type="button" value="Logout" name="B2" LANGUAGE =
"JavaScript" onclick="Window.location.href = 'logout.asp'"></p>
</form>
</TD>
</TR>
</TABLE>

<table border="1" width="100%">
  <tr>
    <td width="20%" align="center"><strong>Nº</strong></td>
    <td width="20%" align="center"><strong>Data</strong></td>
    <td width="20%" align="center"><strong>Autor</strong></td>
    <td width="20%" align="center"><strong>Assunto</strong></td>
    <td width="20%" align="center"><strong>Lido</strong></td>
  </tr>
<%do while not RecSet.Eof%>
  <tr>
    <td width="20%" align="center"><a
href="le.asp?IDmsg=<%=RecSet("IDmsg")%>"><%=RecSet("IDmsg")%
></a></td>
    <td width="20%" align="center"><%=RecSet("DataDaMsg")%></td>
    <td width="20%" align="center"><%=RecSet("De")%></td>
    <td width="20%" align="center"><%=RecSet("Assunto")%></td>
    <td width="20%" align="center">
    <%Session("Lido")=RecSet("Lido")%>
    <%if Session("Lido") = 0 then%>
          <%="Não"%>
    <%else%>
```

```
        <%="Sim"%>
   <%End if%>
   </td>
  </tr>
  <%RecSet.MoveNext%>
<%loop%>
</table>
<% wm.close %>

<hr>

<p><small><font color="#000080">Copyright © 1998 Daniel Wander
Ribeiro Pereira<br>
Instituto de Tecnologia ORT<br>
Última Atualização: 15/10/1998</font></small></p>
</body>
</html>
```

Comentários:

O primeiro item a ser observado, é logo no início do script, onde é montado um esquema de segurança. Ele simplesmente verifica se houve um login válido no sistema, evitando que um usuário mais "esperto" decore o URL da página e acesse as mensagens de outros usuários.

Outra observação é para o esquema de definição de acesso a mensagem. Aquele campo "Idmsg" criado na tabela mensagem é justamente para realizar um controle interno no sistema, permitindo que o usuário chegue até sua mensagem através de um link. É verificado se a mensagem realmente pertence a ele, e somente então será exibido o seu conteúdo.

Dois script's são chamados aqui: "le.asp" e "nova.asp". Vale ainda ressaltar o botão de logout no sistema, invocando outro script, "logout.asp".

Arquivo: le.asp

Descrição:

Esse script permite a leitura de uma mensagem contida no Recordset da página "mensagem.asp".

Código:

```
<%@ LANGUAGE = VBScript %>
<HTML>
<HEAD>
<TITLE>WebMail ORT: Lê Mensagem Recebida</TITLE>
</HEAD>
```

```
<BODY background="fundo_.gif">
<p style="background-color: rgb(255,255,0)"><font
color="#000080"><big><big><strong>Instituto
de Tecnologia ORT (Intranet)</strong></big></big></font></p>

<p align="right"><font
color="#800000"><big><em><big><big><strong>WebMail
ORT</strong></big></big></em></big></font></p>

<hr>

<%
'Declaração e inicialização de variáveis
Dim NomeDoUsuario, IDmsg

IDmsg = request("IDmsg")
NomeDoUsuario = Session("NomeDoUsuario")

'Conexão com o Banco de Dados
Set wm = Server.CreateObject("ADODB.Connection")
DataSource = "webmail"
wm.Open DataSource, "admin", ""

'Procura pela mensagem
Comando = "SELECT * FROM Mensagens WHERE IDmsg = " &
Cdbl(IDmsg)
Set RecSet = wm.Execute(Comando)
if not RecSet.Eof then
Comando = "UPDATE Mensagens SET Lido = True WHERE IDmsg = " &
Cdbl(IDmsg)
        wm.Execute(Comando)
End if
%>

<%if Not RecSet.Eof then%>

<form>
        <input language="JavaScript" type="button" value="Responder"
        onclick="Window.location.href =
'responde.asp?IDmsg=<%=IDmsg%>'" name="Responder">
        <input language="JavaScript" type="button" value="Excluir
Mensagem"
        onclick="Window.location.href =
'apaga.asp?IDmsg=<%=IDmsg%>'" name="Exclui">
<input language="JavaScript" type="button" value="Voltar para
Mensagens"
        onclick="Window.location.href = 'mensagem.asp'"
name="Voltar">
</form>
```

```
<TABLE BORDER=0>
<TR>
	<TD><B>De:</B></TD><TD><%=RecSet("De")%></TD>
</TR>
<TR>
	<TD><B>Para:</B></TD><TD><%=RecSet("Para")%></TD>
</TR>
<TR>
	<TD><B>Assunto:</B></TD><TD><%=RecSet("Assunto")%></
TD>
</TR>
<TR>
<TD COLSPAN=2><B>Conteúdo:</B></TD><TR><TD
COLSPAN=2><%=RecSet("Conteudo")%></TD></TR>
</TR>
</TABLE>
<%Else%>
	<H3><B>Erro:</B> Não foi encontrada a mensagem!</H3>
<%End if%>
<hr>

<p><small><font color="#000080">Copyright © 1998 Daniel Wander
Ribeiro Pereira<br>
Instituto de Tecnologia ORT<br>
Última Atualização: 15/10/1998</font></small></p>

</BODY>
</HTML>
```

Comentários:

A primeira tarefa desse script, é mudar o estado da mensagem, de "não lida" para "lida". Isso é feito assim que se entra na página, onde ela selecionada e um recordset é criado e em seguida atualizado o valor do campo "Lido".

Também nesse script, encontramos os comandos de exclusão da mensagem e de resposta ao autor.

O resto é simplesmente a apresentação dos dados contidos nos campos relativos a mensagem selecionada.

Arquivo: nova.asp

Descrição:

Essa é a página responsável por enviar mensagens através do serviço de WebMail.

Código:

```
<%@ LANGUAGE=VBScript %>
<%'Proteção para impedir acesso direto:
if Session("Senha") = "" then
        response.redirect "index.htm"
End if
%>
<html>

<head>
<title>WebMail ORT: Nova Mensagem</title>
<SCRIPT LANGUAGE="VBScript">
Sub Hoje()
        document.f1.DataDaMsg.value = Date()
End Sub
</SCRIPT>
</head>

<body background="fundo_.gif" onLoad = "Call Hoje()">

<p style="background-color: rgb(255,255,0)"><font
color="#000080"><big><big><strong>Instituto
de Tecnologia ORT (Intranet)</strong></big></big></font></p>

<p align="right"><font
color="#800000"><big><em><big><big><strong>WebMail
ORT</strong></big></big></em></big></font></p>

<hr>

<p><font color="#000080"><em><big><big><strong>Nova
Mensagem</strong></big></big></em></font></p>

<form method="POST" action="novamsg.asp" name = "f1">
  <input type="submit" value="Enviar" name="Enviar"><input
type="reset" value="Cancelar"
  name="Cancelar">
  <input language="JavaScript" type="button" value="Voltar para
Mensagens"
  onclick="Window.location.href = 'mensagem.asp'"
name="Voltar"><BR>
  <p>De:<input type="text" name="De" size="20" VALUE =
<%=Session("NomeDoUsuario")%>>    Para:<input
type="text"
  name="Para" size="20"><br>
  Assunto:<br>
  <input type="text" name="Assunto" size="48"><br>
  Conteúdo:<br>
  <textarea rows="6" name="Conteudo" cols="47">Se desejar, poderão
ser usados os comandos de formatação de texto do padrão HTML.
Apague esse texto.</textarea><br>
  <input type="hidden" name="DataDaMsg" size="20"><br>
```

```
 </p>
</form>

<hr>

<p><small><font color="#000080">Copyright © 1998 Daniel Wander
Ribeiro Pereira<br>
Instituto de Tecnologia ORT<br>
Última Atualização: 15/10/1998</font></small></p>
</body>
</html>
```

Comentários:

Como é uma tarefa crítica, pois deve ser dado um nome ao remetente (não é possível enviar mensagens anônimas), é verificado primeiramente se houve um login válido no sistema.

Ao terminar de preencher o formulário, o usuário envia a mensagem, assim, executando o script “novamsg.asp”.

Arquivo: novamsg.asp

Descrição:

Essa é a página responsável gravar uma nova mensagem no banco de dados.

Código:

```
<%@ LANGUAGE = VBScript %>
<HTML>
<HEAD>
<TITLE>WebMail ORT: Mensagem Enviada</TITLE>
</HEAD>
<BODY background="fundo_.gif">
<p style="background-color: rgb(255,255,0)"><font
color="#000080"><big><big><strong>Instituto
de Tecnologia ORT (Intranet)</strong></big></big></font></p>

<p align="right"><font
color="#800000"><big><em><big><big><strong>WebMail
ORT</strong></big></big></em></big></font></p>

<hr>

<%
'Declaração e inicialização de variáveis
Dim NomeDoUsuario, DataDaMsg, De, Para, Assunto, Conteudo

NomeDoUsuario = Session("NomeDoUsuario")
```

```
DataDaMsg = request.form("DataDaMsg")
De = request.form("De")
Para = request.form("Para")
Assunto = request.form("Assunto")
Conteudo = request.form("Conteudo")

'Conexão com o Banco de Dados
Set wm = Server.CreateObject("ADODB.Connection")
DataSource = "webmail"
wm.Open DataSource, "admin", ""


'Procura pelo destinatário
Comando = "SELECT * FROM Usuarios WHERE NomeDoUsuario = '" &
trim(Para) & "'"
Set RecSet = wm.Execute(Comando)

if Not RecSet.Eof then
        'Insere a nova mensagem
        Comando = "INSERT INTO Mensagens (NomeDoUsuario,
DataDaMsg, De, Para, Assunto, Conteudo)" & "VALUES('" & Para & "','"
& DataDaMsg & "','" & De & "','" & Para & "','" & Assunto & "','" &
Conteudo & "')"
        wm.Execute(Comando)
        response.write "<center><table
border=3><tr><td><H3>Mensagem
Enviada!</H3></td></tr></table></center>"
else
        response.write "<H3>O destinatário não existe!</H3>"
End if

%>

<form>
<div
  align="center"><center><p><input language="JavaScript"
type="button" value="Voltar para Mensagens"
  onclick="Window.location.href = 'mensagem.asp'" name="Voltar">
</p>
  </center></div>
</form>
<hr>

<p><small><font color="#000080">Copyright © 1998 Daniel Wander
Ribeiro Pereira<br>
Instituto de Tecnologia ORT<br>
Última Atualização: 15/10/1998</font></small></p>

</BODY>
</HTML>
```

Comentários:

Esse script simplesmente grava a nova mensagem através de um instrução SQL e avisa ao usuário que a operação foi bem sucedida. Ele permite que o usuário retorne a página de mensagens, usando o mesmo login.

Arquivo: responde.asp

Descrição:

Nessa página o usuário tem a oportunidade de responder à uma mensagem recebida, incluindo automaticamente o nome do destinatário e um trecho da mensagem que ele enviou.

Código:

```
<%@ LANGUAGE = VBScript %>
<HTML>
<HEAD>
<TITLE>WebMail ORT: Responder ao Autor</TITLE>
<SCRIPT LANGUAGE="VBScript">
Sub Hoje()
        document.f1.DataDaMsg.value = Date()
End Sub
</SCRIPT>
</HEAD>
<BODY background="fundo_.gif" onLoad = "Call Hoje()">
<p style="background-color: rgb(255,255,0)"><font
color="#000080"><big><big><strong>Instituto
de Tecnologia ORT (Intranet)</strong></big></big></font></p>

<p align="right"><font
color="#800000"><big><em><big><big><strong>WebMail
ORT</strong></big></big></em></big></font></p>

<hr>

<%
'Declaração e inicialização de variáveis
Dim IDmsg

IDmsg = request("IDmsg")

'Conexão com o Banco de Dados
Set wm = Server.CreateObject("ADODB.Connection")
DataSource = "webmail"
wm.Open DataSource, "admin", ""
```

```
'Procura a mensagem
Comando = "SELECT * FROM Mensagens WHERE IDmsg = " &
Cdbl(Idmsg)
Set RecSet = wm.Execute(Comando)

%>

<p><font color="#000080"><em><big><big><strong>Responder
Mensagem</strong></big></big></em></font></p>

<form method="POST" action="novamsg.asp" name = "f1">
  <input type="submit" value="Enviar" name="Enviar"><input
type="reset" value="Cancelar"
  name="Cancelar">
  <input language="JavaScript" type="button" value="Voltar para
Mensagens"
  onclick="Window.location.href = 'mensagem.asp'"
name="Voltar"><BR>
  <p>De:<input type="text" name="De" size="20" VALUE =
<%=Session("NomeDoUsuario")%>>   
  Para:<input type="text" name="Para" size="20"
value=<%=RecSet("De")%>><br>
  Assunto:<br>
  <input type="text" name="Assunto" size="48"
LANGUAGE="JavaScript" Value="Re: <%=RecSet("Assunto")%>"><br>
  Conteúdo:<br>
  <textarea rows="6" name="Conteudo"
cols="47"><%=RecSet("Conteudo")%></textarea><br>
  <input type="hidden" name="DataDaMsg" size="20"><br>
  </p>
</form>

<hr>

<p><small><font color="#000080">Copyright © 1998 Daniel Wander
Ribeiro Pereira<br>
Instituto de Tecnologia ORT<br>
Última Atualização: 15/10/1998</font></small></p>

</BODY>
</HTML>
```

Comentários:

O código ASP acima, é usado para capturar os dados da mensagem anterior e apresenta-los dentro formulário de resposta de maneira que o usuário possa responder ao autor, sem a necessidade de digitar o seu nome e/ou parte da mensagem recebida, para lembrar do assunto tratado. É interessante que o título já aparece preenchido com a as letras “Re:” indicando uma resposta.

Arquivo: apaga.asp

Descrição:

Essa é a página pela exclusão da mensagem corrente (a que o usuário está visualizando).

Código:

```
<%@ LANGUAGE = VBScript %>
<HTML>
<HEAD>
<TITLE>WebMail ORT: Mensagem Ecluída</TITLE>
</HEAD>
<BODY background="fundo_.gif">
<p style="background-color: rgb(255,255,0)"><font
color="#000080"><big><big><strong>Instituto
de Tecnologia ORT (Intranet)</strong></big></big></font></p>

<p align="right"><font
color="#800000"><big><em><big><big><strong>WebMail
ORT</strong></big></big></em></big></font></p>

<hr>

<%
'Declaração e inicialização de variáveis
Dim IDmsg

IDmsg = request("IDmsg")

'Conexão com o Banco de Dados
Set wm = Server.CreateObject("ADODB.Connection")
DataSource = "webmail"
wm.Open DataSource, "admin", ""

'Exlui a mensagem
Comando = "DELETE * FROM Mensagens WHERE IDmsg = " &
Cdbl(Idmsg)
wm.Execute(Comando)

%>
<center>
<table border=3>
        <tr>
        <td><H3>Mensagem excluída!</H3></td>
        </tr>
</table>
```

```
</center>

<form>
<div
  align="center"><center><p><input language="JavaScript"
type="button" value="Voltar para Mensagens"
  onclick="Window.location.href = 'mensagem.asp'" name="Voltar">
</p>
  </center></div>
</form>

<hr>

<p><small><font color="#000080">Copyright © 1998 Daniel Wander
Ribeiro Pereira<br>
Instituto de Tecnologia ORT<br>
Última Atualização: 15/10/1998</font></small></p>

</BODY>
</HTML>
```

Comentários:

Nessa página é executado um comando SQL para excluir a mensagem corrente e em seguida o script ASP apresenta a resposta ao usuário, informando que a mensagem foi excluída.

Arquivo: logout.asp

Descrição:

A função desse script é simplesmente realizar o logout do usuário no sistema WebMail.

Código:

```
<%@ LANGUAGE=VBScript %>
<%
        Session.Abandon
        response.redirect "index.htm"
%>
```

Comentários:

O código acima simplesmente finaliza a sessão berta pelo usuário e apresenta a Home Page do sistema WebMail.