CEETEPS

Centro Estadual de Educação Tecnológica "Paula Souza"
Mantido pelo Governo do Estado de São Paulo
FATEC – Faculdade de Tecnologia de Americana

BANCO DE DADOS
VIA WEB

1º EDIÇÃO

AMERICANA
2000

Dedicatória

Dedico ao meu avô, *in memorian*,
Meu amigo e companheiro.

Agradecimentos

Gostaria de agradecer todos que colaboram para que eu escreve-se um tema tão atual. Especificamente, Claudia Gomes Mealha e Rodrigo Araujo, amigos de trabalho pelo apoio dado. Também agradeço aos amigos de faculdade pela inspiração e sugestões. E especialmente ao Professor Antoni Alfredo Lacerda, meu mentor e amigo.

Em nota pessoal agradeço minha família, pelo incentivo e a minha namorada pelo apoio e paciência às minhas ausências pela elaboração deste trabalho.

Sumário

Lista de Figuras

Resumo

Esta monografia apresenta como desenvolver aplicações Internet Cliente/Servidor com enfoque em banco de dados, mostrando o funcionamento de um CGI, arquitetura Cliente/Servidor e a introdução ao funcionamento das ferramentas mais conhecidas e usadas hoje na internet : Cold Fusion e ASP para o ambiente Microsoft Windows e Perl para o ambiente Unix/Linux. Bem como a conexão de cada ferramenta com um banco de dados realizando tarefas básicas de inserção, pesquisa, alteração e exclusão. Trazendo também um exemplo prático de uma aplicação em ASP.

Abstract

This monograph shows how to develop Internet Client/Server applications focused on database, showing how a CGI works, architecture Client/Server and the introduction of how the well-known and used tools work in the Internet nowadays: Cold Fusion and ASP for Microsoft Windows environment and Perl for Unix/Linux environment. It also shows the connection between each tool and a database carrying out basic tasks of insertion, research, alteration and exclusion, and a practical example of an application in ASP.

1.Introdução

Por que banco de dados via Web?

Desde o surgimento da internet em 1995, no Brasil, vem crescendo cada vez mais o número de sites na internet. Site's que até meados de 1998 eram, em sua maioria, totalmente estáticos onde internauta apenas via os conteúdos existentes em cada link. Sendo que a atualização desses sites eram feitos manualmente pelo webmaster ou o desenvolvedor do site.

Para atualização era preciso alterar o conteúdo desejado localmente na pagina HTML, para depois ser feito o upload via FTP onde se encontrava o site, ficando depois desses passos o site atualizado via internet.

Só que o conteúdo desses "sites" começaram a aumentar, a velocidade nas atualizações tinha que ser mais rápido, já que esse processo era demorado.

Exemplo: Como disponibilizar 1500 produtos de uma empresa na internet com seus respectivos valores, os quais podem variar diariamente. Do modo como vinham sendo feitos os sites o webmaster ou desenvolvedor do site teria que digitar tudo isso, ou montar uma equipe para fazer isso. E como alterar os valores diariamente ?.

Começa então a surgir a necessidade de disponibilizar informações principalmente de empresas via Web, informações de grande conteúdo. Ou seja é necessário ligar o banco de dados da empresa na internet. Sendo que qualquer alteração feita no banco de dados na empresa já feita automaticamente via internet. Já que é a mesma base

Então começam a aparecer no mercado novas ferramentas que possibilitaram e facilitaram a ligação de uma pagina HTML com um banco de dados. Mudando dessa forma os caminhos de desenvolvimento via Internet.

Surgem:

- Fóruns de Discussão

- Centralização de bases de dados de empresas localizadas geograficamente em lugares diferentes (cidade/estado/país).
- Sistemas via Web
- Lojas Virtuais
- Leilões Virtuais

Dentro dessas novas ferramentas de desenvolvimento aparece também antigas ferramentas que com a internet ganharam novas aplicações, como o Perl para o ambiente Unix. E dessas ferramentas de desenvolvimento eu escolhi três as quais são mais usadas hoje na internet, sendo Microsoft Active Server Page, Allaire Cold Fusion e Perl. E todos os scripts (código) criados nessas ferramentas são baseados no conceito de Common Gateway Interfce (CGI).

2. Intrcdução ao CGI

2.1 O que é CGI

CGI é uma ferramenta utilizada para criar e gerenciar sites Web, a qual faz a comunicação entre um servidor Web e outros programas que estejam sendo executados na mesma máquina, criando aplicações interativas para os usuários. CGI não é uma linguagem ou um protocolo, é apenas um conjunto de variáveis e convenções comuns para a passagem e a tradução de informações do servidor para o cliente e vice-versa, através de um ou mais programas. Estes programas são geralmente chamados por uma página Web. Por sua vez, uma página Web utiliza geralmente formulários HTML para iniciar um programa CGI.

CGI está bastante relacionada à Linguagem de Instruções de Hipertexto – HTML – usada para escrever páginas para a WWW. Ela utiliza muitas vezes trechos desta linguagem em seus próprios scripts, a fim de que seja possível construir páginas através dos programas CGI.

2.2 Para que serve ?

A CGI é utilizada quando um servidor Web chama um programa externo, a fim de transmitir dados específicos do usuário para o programa (como a partir de qual host o usuário está conectando ou a entrada que o usuário forneceu através de um formulário HTML). Então o programa processará esses dados e o servidor transmitirá de volta os resultados e a resposta do programa para o browser (Fig.1). Sendo assim, a CGI permite que as páginas Web sejam criadas de maneira dinâmica, baseada na entrada dos usuários, o que não pode ser obtido apenas com a linguagem HTML.

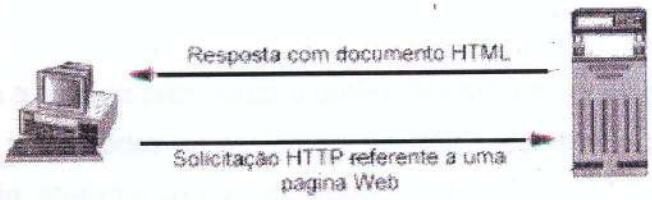

Fig 1

2.3 Onde é usada ?

É possível utilizar os scripts CGI para criar uma grande faixa de aplicações, desde exames até ferramentas de pesquisa, fóruns, murais, salas de bate papo, leilões, lojas virtuais dentre outras aplicações. É possível contar o número de usuários com acesso a um documento ou permitir que eles assinem um livro de convidados eletrônico. Pode-se também fornecer aos usuários todos os tipos de informações, reunir seus comentários e responder-lhes.

Podemos citar um exemplo prático. Suponha que você crie um livro de convidados para seu site Web. A página do livro de convidados pedirá que os usuários subscrevam seu primeiro e último nomes usando um formulário de preenchimento composto por dois campos de entrada de textos (Fig. 2).

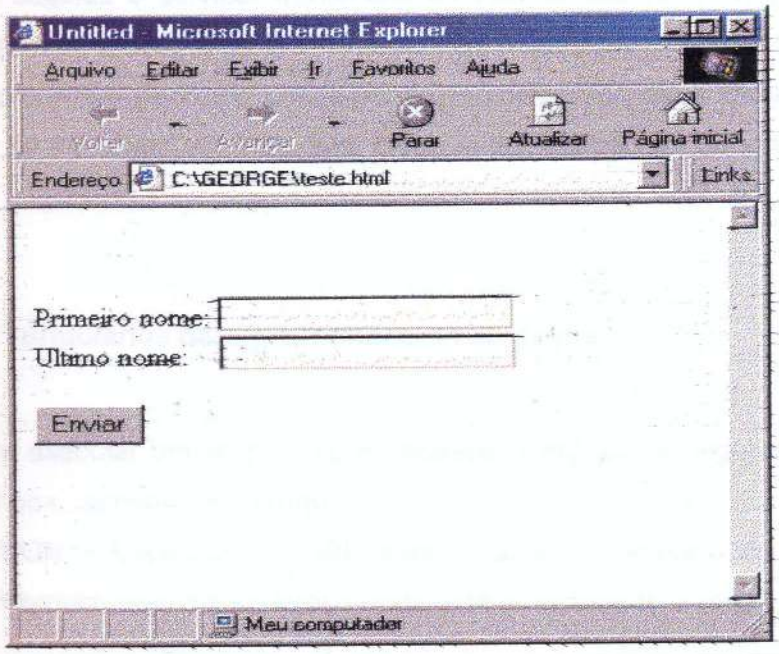

Fig. 2

Quando o usuário pressionar o botão "enviar", os dados fornecidos nos campos de texto serão transmitidos para o programa CGI especificado pelo atributo action da tag <form>. Veja, abaixo o código do arquivo da Fig 2.

```
<Form METHOD="POST" ACTION="teste.htm">
Primeiro nome: <INPUT NAME="nome">
<br>
Ultimo nome: <INPUT NAME="Ultimo Nome">
<br>
<INPUT TYPE="submit"VALUE="Enviar">
<INPUT TYPE="reset"VALUE="Apagar">
</form>
```

2. 4 Transferindo os dados do formulário para a CGI

Os parâmetros para um programa CGI podem ser transferidos no URL ou no texto do corpo da solicitação. O método usado para transmitir os parâmetros é determinado pelo atributo method na tag <form>. O método GET diz para transferir os dados no próprio URL enquanto o método POST diz para usar a parte do corpo da solicitação HTTP para transmitir os parâmetros.

Em seguida o servidor transmitirá os valores das variáveis para o programa CGI. A partir daí, o programa CGI recupera as informações de maneira apropriada e então as processa, podendo utiliza-las de inúmeras maneiras desde simplesmente compilar o nome obtido em uma lista, como pode retornar um anagrama do nome do usuário.

2.5 Formulários para preenchimento de dados

Para executar um script CGI, é necessário especificar alguns atributos do tag <form method=...action=...>...</form>.

ACTION - Especifica o URL para o qual o conteúdo do formulário será mandado. Normalmente é o endereço onde está armazenado o script.

METHOD - Indica como é feita a passagem de dados GET ou POST .

GET: Neste método, os dados de entrada são armazenados pelo servidor antes da execução do script. O script então, acessa essa variável e obtém seu conteúdo. Na prática o método GET coloca os parâmetros digitados pelo usuário no final da URL, no caso do exemplo:

```
http://www.dominio.com.br/cgi-bin/test-
cgi?name1=conteudo1&name2=conteudo2
```

Este método é usado quando não se tem muitos valores de entrada. Também pode ser usado com a inclusão direta do parâmetro no URL não usando <FORM> tag.

```
http://www.domino.com.br/cgi-bin/test-cgi?name1=João&name2=Silva
```

POST: Neste método, o servidor coloca os dados na entrada padrão do sistema operacional e deixa a variável de ambiente livre para outras aplicações.

Outra vantagem do POST em relação ao GET é que não há limite para o tamanho dos dados de entrada. Assim no caso de entradas longas, somente é possível utilizar POST, já que a variável de ambiente tem limite de comprimento.

2.6 Linguagem de programação

Programas em CGI podem ser escritos em qualquer linguagem. Os únicos critérios para a linguagem são os seguintes:

- A linguagem deve ser aceita pelo sistema operacional em que o servidor esta rodando;
- A linguagem deve Ter facilidades suficientes para realizar a tarefa que você precisa realizar com ela;
- O programador deve estar acostumado o bastante com a linguagem para codificar com habilidade.

A maioria dos servidores Web hoje em dia roda algum tipo de UNIX, portanto as linguagens mais utilizadas e disponíveis são C e Perl. Em servidores Microsoft as mais usadas são ASP e Cold Fusion. Portanto, a linguagem deve ser escolhida de acordo com seu ambiente e aplicação.

2.7 Estrutura de um programa CGI

Os programas em CGI geralmente apanham sua entrada de variáveis de ambiente e enviam sua saída para a saída padrão (stdout). A saída do programa precisa estar em um formato que o browser Web possa exibir. Em geral esse formato é o padrão HTML.

Primeiramente, o programa deve dizer ao browser a que tipo de dado ele se refere. Isso é feito com a seguinte diretiva:

```
Content-type: <tipo-MIME>
```

Content-type especifica o tipo MIME do fluxo de informações que está se enviando para o usuário. Geralmente, este será "text/html" para documentos HTML ou "text/plain" para documentos informativos. Outros tipos podem ou não ser aceitos pelo browser do usuário, e por isso devem ser usados com cuidado.

Os cabeçalhos MIME sempre devem ser separados do corpo de um documento por uma linha em branco – isso faz parte do padrão MIME. Não é necessário usá-lo.

A próxima parte do programa exibe todas as variáveis que fazem parte do padrão CGI e as instruções necessárias para processar as informações.

Para rodar o programa, este deve estar no servidor em um diretório adequado, e deve também, ser referenciado por um browser. Um programa em CGI possui três funções de I / O (entrada e saída) básicas. São elas:

- Reunir entrada do servidor, na forma de variáveis padronizadas, dados de formulário e dados de consulta;
- Fornecer dados de saída para o cliente (o browser Web);

- Fornecer informações de negociação de conteúdo (o cabeçalho MIME) para o servidor e o cliente.

Em CGI, a maioria da entrada é passada em variáveis ambiente que são definidas pelo servidor HTTP. Algumas dessas variáveis são padronizadas na especificação CGI; outras são particulares aos browsers, servidores, sites ou outros fatores.

Quando o programa CGI for chamado, diversas variáveis de ambiente já estarão preparadas para lhe dar informações sobre o usuário, sua configuração de software e o ambiente do servidor.

2.8 O ambiente

Os programas CGI são executados em um ambiente especial. O servidor WWW que iniciou o programa CGI cria algumas informações especiais para o mesmo e também espera algumas respostas especiais vindas do programa CGI. Antes de ser iniciada a execução do programa CGI, o servidor da WWW já criou um ambiente especial de processamento para a operação deste programa. O ambiente inclui a tradução de todos os cabeçalhos de solicitação de HTTP para variáveis de ambiente, as quais podem ser utilizadas pelo programa CGI para todos os tipos de informações úteis. Além de informações de sistema (como a data atual), o ambiente inclui informações sobre quem está chamando o programa CGI, de onde o programa está sendo chamado e possivelmente até mesmo informações de estado que ajudem a controlar as ações de cada visitante da Web.

Em seguida, o servidor tenta determinar que tipo de arquivo ou programa está sendo chamado, pois deve atuar de forma diferente de acordo com cada tipo de arquivo acessado. Portanto, o seu servidor da WWW primeiro avalia a extensão do arquivo para determinar se será preciso analisar o arquivo à procura de comandos, executar o interpretador de determinada linguagem para compilar e executar o programa ou apenas gerar os cabeçalhos de resposta corretos de HTTP e retornar um arquivo da HTML.

Após o servidor ter iniciado um programa CGI ou até mesmo um arquivo HTML, será esperado um tipo específico de resposta vindo do programa CGI. Se o servidor estiver retornando apenas um arquivo HTML, será esperado que esse arquivo seja de

tipo texto, contendo tags de HTML e textos. Se o servidor estiver retornando um arquivo HTML, o próprio servidor será responsável pela geração de cabeçalhos de resposta requeridos pelo HTTP, os quais informam ao navegador que executou a chamada, o status da solicitação desse navegador para uma página da Web e que tipo de dados será recebido pelo navegador, entre outras coisas.

Se o servidor identificar o arquivo como um programa executável CGI, este será executado de forma apropriada. Após a execução do programa por parte do servidor, o programa normalmente responde com os cabeçalhos de resposta mínimos do HTTP exigidos e, em seguida, com algumas tags HTML. Se o programa CGI estiver retornando HTML, ele deve gerar um cabeçalho de resposta como Content-Type: text/html. Essa resposta fornece ao servidor a quantidade suficiente de informação para gerar quaisquer outros cabeçalhos de resposta do HTTP exigidos.

Deste modo, o programa CGI, a HTML e o HTTP precisam operar de forma coordenada a fim de que a sua aplicação on-line para a Internet funcione de forma apropriada. O código HTML define a forma como o usuário visualiza a interface do programa, sendo também responsável pela coleta de dados inseridos pelo usuário. Isso é freqüentemente denominado *código da Interface Humana com o Computador;* trata-se da janela onde o programa e o usuário interagem mutuamente. O HTTP representa o mecanismo de transporte para o envio de dados entre o programa CGI e o usuário. Ele é o diretor oculto que traduz e envia as informações entre o cliente da Web e o programa CGI. O programa CGI será responsável por entender as instruções do HTTP e as solicitações do usuário, além de receber as solicitações feitas pelos usuários e retornar respostas válidas e úteis ao cliente da Web que está dando cliques na sua página HTML da Web.

2.9 Servidor Web

Como os scripts CGI são programas externos, é importante, por questões de segurança, manter um controle sobre sua execução e sobre quem está autorizado a implementá-la.

Normalmente, um servidor Web possui três áreas: a principal, a de usuários e a de scripts.

A principal ou raiz é a área a partir da qual todos os outros endereços se relacionam.
A área de usuário é a área em que um usuário pode colocar seus documentos públicos a partir de sua área de trabalho.
Na área de script, somente são mantidos programas executáveis.

3. As Ferramentas e seus ambientes de trabalho

As ferramentas de desenvolvimento para interligação com banco de dados são:

Cold Fusion

Criado pela Allaire permite um fácil interação com banco de dados para ambiente Windows utilizando para conexões drivers ODBC podendo dessa forma se comunicar com qualquer base de dados, como exemplo: MS-SQL Server e ORACLE.

Active Server Page

Desenvolvido pela Microsoft possui a facilidade de integração com banco de dados utilizando para conexão drivers ODBC, podendo ser ligado diretamente a qualquer base de dados como : bases .mdb ou ao um servidor MS-SQL Server ou ORACLE utilizando da tecnologia de conexão ADO (Active Data Object).

Perl

Desenvolvido para ambiente UNIX, é um pouco mais complicado para se usar, sendo que a conexão podem ser feita com My-SQL , Oracle, SyBase através do DBI (Database Interface Module)

4. Cold Fusion

4.1 Conhecendo o Cold Fusion

Desenvolvido pela Allaire Coorporation possui uma linguagem fácil entendimento. Os códigos são embutidos dentro das paginas HTML's através de tag's . O Cold Fusion é composto por dois software's:

- Cold Fusion Editor: Editor HTML integrado com suas tag's CFML
- Cold Fusion Server: Responsável por criar o device que será usado para ligação da aplicação Web com o banco de dados e por interpretar as tags CFML.

Trabalha com os padrões ODBC.

Roda sobre o Internet information Server ou Personal Web Server da Microsoft.

4.2 Vantagens

É o mais simples e fácil de utilizar dos apresentados nessa monografia. Possui o Cold Fusion Editor, editor desenvolvido pela Allaire que facilita a criação de uma aplicação Web.

5. Active Server Page

5.1 Conhecendo o Active Server Page

Desenvolvido pela Microsoft, sua linguagem é bem parecida com a do Visual Basic, possuindo algumas diferenças, sendo a principal delas a de que seu código fonte é interpretado e não compilado (como no Visual Basic) o que lhe deu o nome de Visual Basic Script. Portanto o Active Server Page se utiliza do Visual Basic Script.

Baseado na arquitetura cliente/servidor a Microsoft desenvolveu uma ferramenta que permite ao Programador criar aplicações robustas e adequadas para Web com recursos avançados orientado a banco de dados, trabalha integrado ao código HTML ou separado sendo que a inserção dos comandos tem que ser feita dentro das marcações <% e %> que delimitam o inicio e o fim dos comandos VB Script Os arquivos que contém comandos em VB Script precisa ter especificado na primeira linha o código `<%@ LANGUAGE="VBSCRIPT"%>`.

Portando o ASP foi criado exclusivamente para o uso na internet/intranet, que forçada pela internet e pressão de ferramentas concorrente como Perl (Unix), forçou a Microsoft criar esta ferramenta de desenvolvimento para rodar em seus Servidores Web.

5.2 Vantagens

Por ser dá própria Microsoft possui uma melhor integração com as próprias soluções Microsoft para Internet e banco de dados, tendo assim uma melhor performance.

Possui uma linguagem de fácil uso e entendimento, Você pode usar o Interdev, editor próprio desenvolvido pela Microsoft, que permite uma maior velocidade no desenvolvimento de aplicações para Internet/Extranet.

O Interdev possui as seguintes ferramentas para banco de dados que apresentam vários recursos como:

Data View - Permite verificar todos os seus objetos de banco de dados, incluindo tabelas, modos de exibição, procedimentos armazenados.

Query Designer – Uma ferramenta que permite criar consultas de banco de dados SQL e testar os resultados, tudo visualmente.

Database Designer – Uma ferramenta que permite projetar, criar e manter um banco de dados SQL.

Stored Procedure – Uma ferramenta que permite a edição de procedimentos para o Microsoft SQL Server e Oracle.

5.3 O Interpretador de comando

O próprio Servidor Web da Microsoft, Internet Information Server, inclui o mecanismo ActiveX Server Scripiting que interpreta os comandos do script ASP

5.4 Tipos de Dados e Variáveis

Existe apenas um tipo de dados para uma variável VBScript. A *variante* funciona como o tipo de dados para todas as variáveis criadas no seu Código do VBScript. Uma variante é muito flexível porque pode armazenar quase todos os tipos de informação. Como : strings, números, datas, objetos.

Sua declaração pode ser feita em qualquer parte do código necessitando apenas de ser atribuída um valor através do sinal de igual (=) Exemplo:

```
vData = "14/04/2000"
vFone = 4623303
```

As variáveis consiste em dois tipos de escopo, em nível de procedimento e em nível de script. O escopo em nível procedimento, é as vezes chamado de escopo

local, refere-se as variáveis declaradas em um procedimento. As variáveis de nível de procedimento só podem ser usadas no contexto do procedimento.

O escopo em nível de script se refere as variáveis que são definidas fora do seus procedimentos. As variáveis em nível de script podem ser reconhecidas em todos os seus procedimentos.

Existe dois tipos muito importantes de criação de variáveis , a de Sessão e de Aplicação. As variáveis de Sessão podem ser definidas em qualquer parte do código e seu conteúdo fica armazenado em memória sendo que é criada uma variável de sessão para cada cliente (Fig.3). Sua declaração é feita através da seguinte forma:

```
Session("ValorCompra") = "120,00"
```

As variáveis de aplicação também podem ser definidas em qualquer parte do código sendo que seu valor fica armazenado em memória global no servidor e seu conteúdo é igual para todos os clientes (Fig. 3). Sua declaração é feita da seguinte forma.

```
Application("Valor_Dolar") = 1,22
```

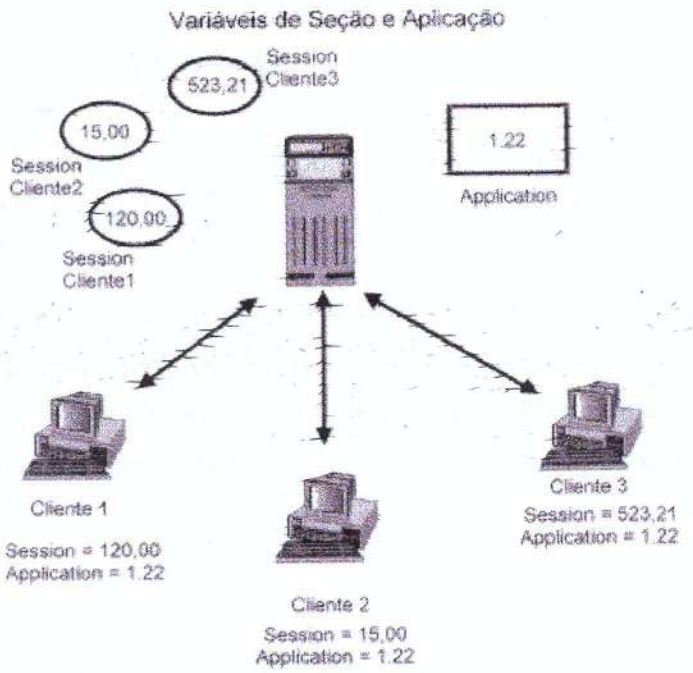

Fig 3

Observe que é criado uma área no Servidor para alocar o conteúdo de cada variável de Sessão de cada Cliente. E que para a variável de aplicação é criado somente uma área, sendo assim todos os clientes tem o mesmo valor dessa variável.

Portanto é bom ter para cuidado criar variáveis de sessão, pois quantos mais Clientes acessando o Servidor mais memória é ocupado na máquina.

5.5 Conexão com banco de dados

Para comunicação com banco de dados a Microsoft projetou o ADO, ActiveX Data Objects, para serem projetos independentes de linguagem para você acessar um banco de dados a partir de paginas Web. Quem programa em Visual Basic está familiarizado com os modelos DAOs (Data Access Objects) e RDOs (Remote Data Access) o ADO é o sucessor desses modelos, pois o ADO combina os melhores métodos de acesso a dados.

O vantagem principal é que o ADO você pode criar objetos independentes de acesso ao banco de dados, tornando o acesso ainda mais rápido.

Veja alguns objetos do ADO:

Objeto Connection – controla a conexão com o banco de Dados

Objeto Command – permite especificar o comando que será executado no banco de dados

Objeto RecordSet – utilizado para manipulação dos registros ou linhas da sua tabela.

Objeto Open – abri o objeto do conexão

Objeto ConectionTimeOut – define o tempo máximo de conexão

Objeto CommandTimeOut – define o tempo máximo de execução de comandos.

A conexão pode ser feita de duas formas:

- Através do ODBC, onde é criado o System DSN (Data Source Name) o qual é ligado a base de dados. Sendo todas as referências a base de dados no ASP é feita pela System DSN criado e este por sua vez se comunicará com o banco.
- Ou direto com a base de dados, sem utilizar DSN ficando dessa forma mais rápido o acesso a base.

Trabalha com os padrões : ODBC, SQL Server, ORACLE

6 Perl

6.1 Conhecendo o Perl

Perl é uma linguagem para manipular números, textos, arquivos, diretórios, computadores, redes e programas. Também para desenvolver, modificar e depurar seus próprios programas. Originalmente designado para o processamento de textos, ele desenvolveu-se em uma linguagem de programação sofisticada com um ambiente de desenvolvimento de software rico e completo com depuradores, perfis, referências cruzadas, interpretadores, bibliotecas, editores direcionados para a sintaxe e todo o resto de armadilhas de uma linguagem de programação.

6.2 Vantagens

Perl é uma linguagem que não requer compilação, além disso, é uma linguagem "grátis" de fácil aprendizado e muito semelhante a linguagem C a qual é, provavelmente, a mais popular do mundo. Perl trabalha no ambiente Unix, facilitando a implementação de todas as chamadas ao sistema, possui recursos internos de segurança e apresenta uma velocidade relativamente alta.

Em complemento, a Perl possui uma excelente estrutura de dados denominada array associativo, a qual pode ser utilizada na manipulação de banco de dados. Um dos recursos internos de segurança é o rasteio do fluxo de dados, o qual nos permite descobrir a origem dos dados inseguros. Essa capacidade freqüentemente impede operações inseguras antes que elas ocorram.

6.3 O interpretador Perl

O executável Perl também é chamado de interpretador Perl. Todo programa Perl, para ser executado, tem que ser transmitido através do interpretador Perl. A primeira linha em muitos programas Perl é algo como:

```
#!/usr/bin/perl
```

Para os sistemas Unix, esta linha "#!" informa ao shell para procurar o programa /usr/bin/perl e transmitir o resto do arquivo a esse programa para sua execução. Algumas vezes, este caminho poderá ser diferente, mas quase todos os programas Perl no Unix começam com alguma variação desta linha.

Se aparecer um erro do tipo "Command not found" em um programa Perl, geralmente será porque o caminho para o executável Perl está errado. Deve-se verificar o caminho correto do local do interpretador Perl.

Então o interpretador Perl compila o programa internamente em uma árvore de análise e então executa-o imediatamente. O Perl é comumente conhecido como uma linguagem interpretada, entretanto o interpretador converte de fato o programa em um código de bytes antes de executá-lo, é algumas vezes chamado de interpretador / compilador.

Então é necessário somente escrever um programa Perl, fornecer-lhe a linha #! correta no início do script, colocar no Servidor Web Unix.

6.4 Tipos de dados e variáveis

As variáveis em Perl, não precisam ser declaradas inicialmente. Elas são criadas automaticamente quando forem atribuídas pela primeira vez. A declaração das variáveis entram em cena quando for preciso limitar o escopo do uso de uma variável. É possível fazer isto de duas maneiras:

- escopo dinâmico cria objetos temporários em um escopo. As construções com escopo dinâmico são visíveis globalmente, mas apenas entram em vigor em seus escopos definidos. O escopo dinâmico aplica-se às variáveis declaradas com local.
- escopo lexical cria construções privadas que são visíveis apenas em seus escopos. A forma vista com mais freqüência de declaração com escopo lexical é a declaração das variáveis my .

É permitido usar o mesmo nome para uma variável escalar, um array ou um hash (isso significa que $aux é diferente de @aux e %aux. A atribuição de variáveis usa o operador (=) com os devidos dados.

Qualquer variável usada em uma função que não seja declarada como privada será uma variável global. Nas sub-rotinas, geralmente se pretende usar variáveis que não serão usadas em nenhum outro lugar no programa e não se deseja que ocupem

memória quando a sub-rotina não estiver sendo executada. É possível também não querer alterar as variáveis nas sub-rotinas que possam ter o mesmo nome das variáveis globais.

A função my declara as variáveis que tem um escopo lexical na sub-rotina. Para colocar no escopo diversas variáveis de uma só vez, use uma lista entre parêntesis. É possível também atribuir uma variável em uma instrução my:

```
my @list = (44,55,66);
my $cd = "orb";
```

Perl tem três tipos de dados básicos: escalares, arrays e hashes.

Escalares

São basicamente variáveis simples, precedidos por um cifrão ($). Um escalar pode ser um número, uma string ou uma referência (uma referência é um escalar que aponta para outra parte dos dados). Se for fornecido uma string onde um número é esperado, ou vice-versa, o Perl converterá automaticamente o operando. Exemplo de escalares:

```
$nome = "João";
$idade = 32;
```

Arrays

Os arrays são listas ordenadas de escalares que é acessada com um subscript numérico (começando em 0), precedidos por um sinal @. Os índices negativos contam de traz para frente a partir do último elemento da lista (por exemplo –1 se refere ao último elemento da lista). Exemplos:

```
@num[0] = 1;
@num[1] = "Um";
```

Hashes

Conjuntos desordenados de pares chave / valor que será acessado usando as chaves como subscripts. São precedidos por um sinal de porcentagem %. Exemplo:

```
%semana{'Seg'} = "Segunda-feira";
```

6.5 Conexão com Banco de Dados

As conexões com base de dados são feitas através do DBI (Data Base Interface) que não sabe conversar em particular com a base de dados mas ele consegue localizar e carregar o DBD (Data Base Driver) que faz a comunicação da base de dados. Ou seja o DBI utiliza módulos DBD de determinada base de dados para adquirir os dados de maneira padronizada. Trabalha de maneira simples, enviando os comandos SQL e retornando os dados de volta.

Podendo assim utilizar o mesmo método para adquirir/excluir/inserir/alterar dados do Oracle, Sybase ou MySQL.

7 Utilizando as ferramentas

7.1 Recebendo dados via Formulário

O recebimento dos dados pelo Cold Fusion, ASP, Perl é feito da mesma forma através das tag's html de formulário. Veja os exemplos:

7.2 No Cold Fusion

Veja como ficaria o arquivo teste.html da Fig 1 se os dados fossem para um script Cold Fusion:

```
<HTML>
<HEAD>
      <TITLE>Teste Cold Fusion </TITLE>
</HEAD>
<BODY>
<form action="incluir_form.cfm" method="POST">
<input type="Text" name="Nome" Value="" size="30">
<input type="Text" name="Rg" Value="" size="12">
<input type="Submit" name="Enviar" value="Enviar">
</form>
<BODY>
</HTML>
```

7.3 No Active Server Page

No ASP o arquivo teste.html da fig. 1 ficaria assim:

```
<HTML>
<HEAD>
      <TITLE>Teste ASP </TITLE>
</HEAD>
<BODY>
<form action="incluir_form.asp" method="POST">
```

```
<input type="Text" name="Nome" Value="" size="30">
<input type="Text" name="Rg" Value="" size="12">
<input type="Submit" name="Enviar" value="Enviar">
</form>
<BODY>
</HTML>
```

7.4 No Perl

No Perl o código do arquivo teste.html seria assim:

```
<HTML>
<HEAD>
     <TITLE>Teste Perl</TITLE>
</HEAD>
<BODY>
<form action="incluir_form.pl" method="POST">
<input type="Text" name="Nome" Value="" size="30">
<input type="Text" name="Rg" Value="" size="12">
<input type="Submit" name="Enviar" value="Enviar">
</form>
</BODY>
</HTML>
```

Através da método POST os dados como Nome e RG são postados para a respectiva pagina indicada no ACTION. Sendo que esses dados serão tratados pelos arquivos indicados.

O método POST transmite os dados no corpo da solicitação HTTP, ficando oculto os dados. Já o método GET transfere os dados na própria URL exibindo seus valores. Exemplo:

```
http://www.dominio.com.br/incluir_form/asp?Nome=CarLos&Rg=14725896
```

Como já foi dito no capitulo 2, uma vantagem do POST em relação ao GET é que não há limite para o tamanho dos dados de entrada. Assim no caso de entradas longas, somente é possível utilizar POST, já que a variável de ambiente tem limite de comprimento.

8. Trabalhando os Dados

8.1 Tratando os dados

Cada ferramenta de desenvolvimento possui uma forma sintática diferente de tratar os dados vindos da cláusula POST ou GET do form. Veja os exemplos:

8.2 Cold Fusion

O recebimento dos dados tem que ser feito dentro de um bloco de comandos do Cold Fusion. Exemplo do código do arquivo incluir_form.cfm:

```
<cfquery name="consulta_alt" datasource="gindice">
SELECT *
FROM cliente
WHERE nome = '#form.Nome#'
</cfquery>
<HTML>
<HEAD>
      <TITLE>Teste</TITLE>
</HEAD>
<body>
</body>
</html>
```

No exemplo o valor digitado no formulário está sendo utilizado para fazer um pesquisa na base de dados. Observe que o valor postado é recuperado pelo comando

```
Nome = '#form.Nome#'
```

8.3 Active Server Page

Através do comando Request.Form é possível receber os dados postado no formulário. Ex:

incluir_form.asp

```
<HTML>
<HEAD>
     <TITLE>Teste ASP</TITLE>
</HEAD>
<BODY>
<%
vNome = Request.Form("Nome")
vRg = Request.Form("Rg")
%>
</BODY>
</HTML>
```

Utilizando esses comandos o conteúdo das variáveis vNome e vRG passam a ter o valor digitado nos seus respectivos campos do formulário. Tendo o controle dos dados postados é possível fazer a conexão com a base de dados e armazenar, pesquisar, apagar os dados postados.

8.4 Perl

Os dados vindos do formulário é armazenado dentro de uma variável de ambiente padrão do Perl chamada de QUERY_STRING. Os dados dos campos são armazenados seqüencialmente separados pelo símbolo &. No nosso caso ficaria assim:

```
QUERY_STRING teria o valor :  Carlos&14725896
```

Dentro do código Perl utiliza-se o comando **SPLIT** para quebrar o conteúdo da QUERY_STRING, e o comando **@** para criar o vetor onde será armazenados os dados

```
@Vet  Split /&/, QUERY_STRING
vNome = $vet[0]
vRg = $vet[1]
```

9 Trabalhando com Banco de Dados

9.1 Cold Fusion.

A conexão com a base de dados é feita indicando o apenas o nome do datasource criado no Cold Fusion Server. Ele se encarregará de fazer toda comunicação com a base de dados. Veja exemplo do codigo:

```
<cfinsert datasource="gindice" tablename="cliente">
<HTML>
<HEAD>
      <TITLE>Untitled</TITLE>
</HEAD>
<center>Cadastro efetuado com sucesso.</center>
</BODY>
</HTML>
```

Observe o datasource gindice, que é criado no Cold Fusion Server, observe o comando cfinsert que informa que os dados postados no formulário são para inserção de um novo registro.

O nome do campo no formulário tem ser igual ao nome do campo na tabela, pois dessa forma o datasource consegue saber onde será gravado campo, postados no formulário, na tabela do seu banco de dados

9.1.1 Inserção

```
<cfinsert datasource="gindice" tablename="cliente">
```

Insere um novo registro na tabela cliente. Observe o datasource gindice, ele é o responsável por fazer toda comunicação com a base de dados.

9.1.2 Pesquisa

```
<cfquery name="consultanomes" datasource="gindice">
SELECT * FROM cliente  WHERE nome = '#form.nomeabc#'
</cfquery>
```

O comando cfquery executa uma consulta. Observe que foi dado o nome de consultanomes a consulta. O resultado dessa consulta pode ser exibida no browser através do comando :

9.1.3 Saída

```
<cfoutput query="consultanomes">
#nome#<br><br>
#endereco#<br>
#cidade#<br>
#cep#<br>
#email#
<img src="#foto#" border=0>
<br>
</cfoutput>
```

O comando cfoutput exibe o conteúdo para o cliente. No caso da Query feita acima consultanomes. Observe texto que está entre os sinal de # é o nome dos campos na tabela do banco de dados.

9.1.4 Alteração

```
<CFUPDATE DATASOURCE="gindice"    TABLENAME="cliente">
```

Atualiza os valores postados no formulário na tabela cliente. Os dados são inseridos na posição da ultima Query feita.

9.1.5 Exclusão

```
<CFQUERY NAME="exclui"     DATASOURCE="gindice">
    DELETE FROM cliente WHERE nome = #form.nome#
</CFQUERY>
```

Dessa forma é deletado o registro que satisfazer a condição.

9.2 ASP

Como já foi mostrado no item 5.5 existe duas formas de se fazer a conexão com o banco de Dados através do modelo ADO, vamos ver os exemplos:

9.2.1 Conexão vai ODBC através de um DSN (Fig.4)

```
Set ObjAcesso = Server.CreateObject("ADODB.Connection")  // Cria o Objeto de
Conexão
ObjAcesso.ConnectionTimeout = Session("default_ConnectionTimeout") // Define
tempo limite de conexão
ObjAcesso.CommandTimeout = Session("default_CommandTimeout") // Define o tempo
limite para execuçao
ObjAcesso.Open Session("default_ConnectionString"),
Session("default_RuntimeUserName"), Session("default_RuntimePassword") // Abri
a base de dados
Set cmdTemp = Server.CreateObject("ADODB.Command") //  Cria Objeto de execução
Set rstClientes = Server.CreateObject("ADODB.Recordset") // Cria o Recordset de
acesso a base
cmdTemp.CommandText = "SELECT CodCliente, Nome FROM d_Clientes ORDER BY Nome"
// Define o comando a ser usado. No caso um Query SQL.
cmdTemp.CommandType = 1 // Define o tipo de comando
Set cmdTemp.ActiveConnection = Acesso // Informa ao objeto (cmdTemp) a utilizar
o  DSN Acesso (Criado no ODBC)  q será usado na conexão
rstClientes.Open cmdTemp, , 1, 3 // Indica para o Recordset (rstClientes) abrir
o objeto de execução ( cmdTemp) . Sendo que o resultado é jogado no recordset
(rstClientes)

rstPedido.Close // fecha o objeto
Set rstPedido = Nothing // Retira o objeto da memória
```

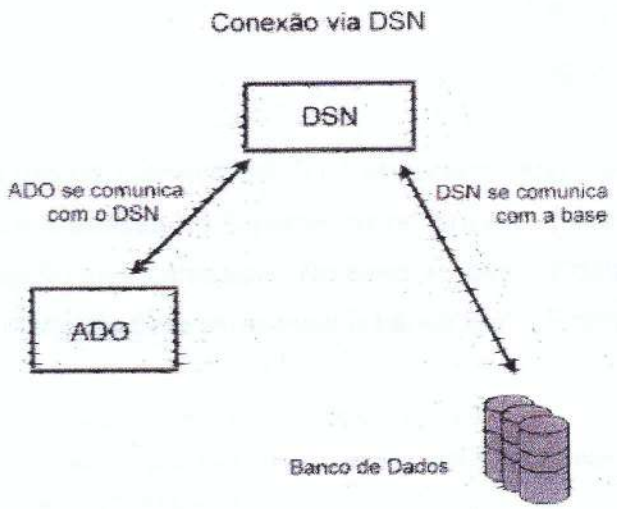

Fig. 4

9.2.2 Conexão direta com a base sem a necessidade de se utilizar um DSN.

O ADO possui uma vantagem que fazer o acesso direto ao banco de dados sem precisar criar um DSN (Fig. 5). Através do código abaixo.

```
Set rstClientes = Server.CreateObject("ADODB.Recordset")  //  Cria o objeto de
manipulação do registro que ira' receber o resultado da Consulta SQL
rstClientes.Open " SELECT CodCliente, Nome FROM d_Clientes ORDER BY Nome'  //
Abre o objeto de conexão e passa os parâmetros da consulta SQL, String de
conexão e tipos de acesso.
Session("default_ConnectionString"), adOpenKeyset, adLockOptimistic, adCmdText
```

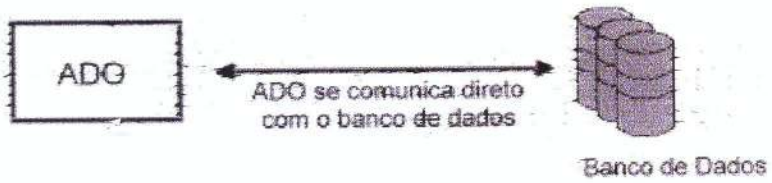

Fig.5

Observe o uso da Session("default_ConnectionString") que pode ser definida no primeiro script que é acesso no sistema ou no arquivo global.asa que é executado antes que a aplicação seja carregada. No caso a Session("default_ConnectionString") seria o comando completo para se acessar o banco SQL.. Exemplo:

```
Session("default_ConnectionString") = "DRIVER=SQL
Server;UID=usuario;PWD=senha;Regional=Yes;DATABASE=Clientes;WSID=NomedoServidor
;APP=Nome da Aplicacao;SERVER=(local)"
```

9.2.3 Inserção

```
<%
Set rstInclui = Server.CreateObject("ADODB.Recordset")
rstInclui.Open "SELECT * FROM clientes WHERE CODID=" & -1,
Session("default_ConnectionString"), adOpenKeyset, adLockOptimistic, adCmdText
rstInclui.AddNew
rstInclui("Nome") = Request.Form("Nome")
rstInclui("Rg") = Request.Form("RG")
rstInclui.Update
%>
```

Observe que o Select posiciona a tabela clientes no final, o comando rstInclui.AddNew permite a inserção de um novo registro e o comando rstInclui.Update grava os dados na tabela. Observando que o texto digitado dentro do rstInclui é o nome do campo na tabela clientes.

9.2.4 Pesquisa

```
<%
vNome = Request.Form("Nome")
Set rstPesquisa = Server.CreateObject("ADODB.Recordset")
rstPesquisa.Open "SELECT * FROM clientes WHERE Nome=" & vNome,
Session("default_ConnectionString"), adOpenKeyset, adLockOptimistic, adCmdText
intQtdRegs = rstPesquisa.RecordCount

if intQtdRegs = 0 then
     Response.write "<center>Nada foi encontrado</center> "
   Else
     Response.write "Codigo: " & rstPesquisa("Nome")
     Response.write "Codigo: " & rstPesquisa("Rg")
End If

%>
```

Observe que o valor digitado no formulário para pesquisa é passado para a variável vNome, que por sua vez é utilizada no select para pesquisar. O comando rstPesquisa.RecordCount retorna para a variável intQtdRegs o numero de registros que satisfizeram a condição. O comando Response.write imprime no cliente os dados, sendo que o comando rstPesquisa("Nome") retorna o valor contido no campo *nome* da tabela cliente.

9.2.5 Saída

```
<%  Response.write "Codigo: " & rstPesquisa("Nome") %>
```

A saída de dados é feita através do comando Response.write, onde no exemplo será impresso no browser cliente o texto Codigo e logo após o valor do campo Nome. Sendo que será impresso o valor de determinado campo através do posicionamento RecordSet rstPesquisa na base, indicado por um instrução SELECT

9.2.6 Alteração

```
<%
Set rstAltera = Server.CreateObject("ADODB.Recordset")
rstAltera.Open "SELECT * FROM clientes WHERE Nome=" & Request.Form("Nome"),
Session("default_ConnectionString"), adOpenKeyset, adLockOptimistic, adCmdText
rstAltera("Nome") = Request.Form("Nome")
rstAltera("Rg") = Request.Form("Disponivel")
rstAltera.Update
%>
```

Observe que a forma de alteração é igual a da inclusão, onde é omitido apenas o comando AddNew. Observe também que ao invés de jogar o conteúdo do valor postado no formulário em uma variável para ser usada como chave no SELECT, podemos informar direto no Select.

9.2.7 Exclusão

```
<%
Set rstAltera = Server.CreateObject("ADODB.Recordset")
rstAltera.Open "DELETE FROM clientes WHERE Nome=" & Request.Form("Nome"),
Session("default_ConnectionString"), adOpenKeyset, adLockOptimistic, adCmdText
%>
```

O registro que satisfizer a condição é apagado.

9.3 Perl

A conexão com banco de dados é um pouco mais complicada em Perl, seguindo o padrão abaixo:

```
$dbh = DBI->connect('<usuario>','<tabela>') or Die "Impossivel conectar. Erro
$DBI::db_errstr";
```

Neste trecho, conectamos ao banco de dados. Definimos o usuário que já foi criado no próprio banco de dados que possui acesso e o nome da tabela. A variável $DBI::db_errstr retorna o tipo de erro caso não seja possível fazer a conexão. Veja abaixo um exemplo de conexão com o Banco de Dados MySQL.

```
$dbh = Mysql->connect('carlos','clientes') or Die "Impossivel conectar. Erro
$MySQL::db_errstr";
```

Logo após a conexão definimos então o comando SQL a ser executado. O padrão de definição é a seguinte:

$sth = $dbh->query("$query");

9.3.1 Inserção

Para fazer uma inclusão define-se a query:

```
$query = "INSERT INTO clientes (Nome,Rg,Telefone) VALUES ($Nome,$Rg,$Telefone)"
```

Onde (Nome,Rg,Telefone) são os campos da tabela e ($Nome,$Rg,$Telefone)" são as váriaveis de programa.

9.3.2 Pesquisa

Para fazer uma inclusão define-se a query:

```
$query = "SELECT * FROM clientes WHERE Cliente=$varCli"
```

9.3.3 Saída

Para imprimir os dados o código é:

```
For ($loop = 1; $loop <= $sth->numrows; loop++)  # Faz um loop do registro 1
até o ultimo retornado da query
{
      ($tmp) = $sth->fetchrow;  # Pega a linha corrente da query e joga na
variável $tmp
      $dados[$loop] = $tmp;  # Pega o valor de $tmp e joga em uma posição da
matriz @dados
}
```

9.3.4 Alteração

Para fazer uma atualização utiliza-se

```
$query = "UPDATE Alunos SET nota=7 WHERE nota BETWEEN 7 and 6.5"
```

Nesse exemplo é feita uma atualização onde todos os alunos que tiverem nota entre 7 e 6.5 receberam nota 7.

9.3.5 Exclusão

Para fazer uma exclusão utiliza-se

```
$query = "DELETE FROM Clientes WHERE Rg=$vrg"
```

9.3.6 Exemplo

Veja o exemplo completo de uma conexão com banco de dados, a execução da query e a impressão do resultado.

```
use DBI; # Carrega o Módulo DBI
# Informa ao DBI qual o tipo de base de dados será feira a conexão
# No caso Oracle e a tabela é a Clientes
my $dbh = DBI->connect('DBI:Oracle:clientes') or
die "Impossível conectar à base" . DBI->errstr;
my $sth = $dbh->prepare('SELECT * FROM people WHERE lastname = ?') or
die "Couldn't prepare statement: " . $dbh->errstr; print "Enter name> ";
while ($lastname = <>) { # Lê informações do usuário
my @data;
        chomp $lastname;

        $sth->execute($lastname) # Executa a query

        or die "Erro ao executar a query" .

        $sth->errstr;

        # Lê os registros e imprime-os na tela.

        while (@data = $sth->fetchrow_array()) {

        my $firstname = $data[1];

        my $id = $data[2];

        print "\t$id: $firstname $lastname\n";

        }

        if ($sth->rows == 0) {

        print "No names matched `$lastname'.\n\n";
} print "\n"; print "Enter name> ";
}
$sth->finish; #
$dbh->disconnect; # Finaliza a conexão com a base de dados
```

Esse código faz a conexão com banco de dados, lê as informações que atendem a Query em seguida imprime os resultados.

Observe a instrução SELECT especificada no *prepare* sendo que para inclusão, pesquisa, alteração e exclusão é utilizado da mesma forma.

10 Projeto em ASP

Noticias Online consiste numa aplicação onde a usuário cadastra uma noticia e esta passa vigorar automaticamente em um site da Web, utilizando um banco Microsoft Access como banco de dados.

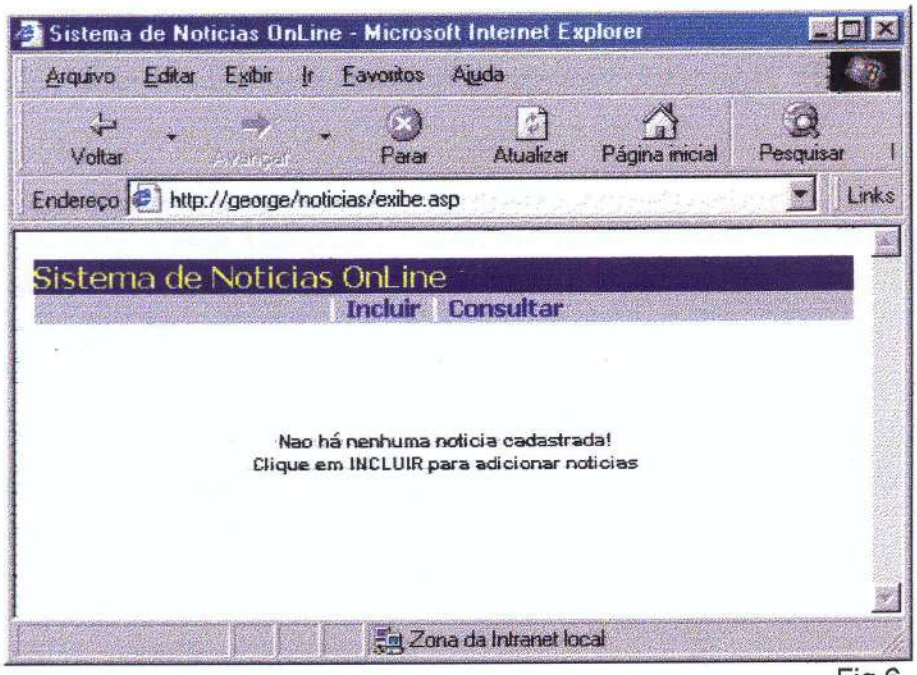

Fig 6

A figura 6 mostra o arquivo exibe.asp já interpretado pelo Browser. Que tem a função de listar as noticias já castradas pelo usuário.

O Código do arquivo exibe.asp é:

```
<%@ LANGUAGE="VBSCRIPT"%>
<HTML>
<HEAD>
      <TITLE>Sistema de Noticias OnLine</TITLE>
</HEAD>
<BODY bgcolor="White">
<table width=100% cellspacing="0" cellpadding="0">
<tr>
```

```
      <td bgcolor="Navy"><font face="Tahoma" size="3"
color="Yellow"><b>Sistema de Noticias OnLine</b></font></td>
</tr>
<tr>
      <td align="CENTER"  bgcolor="Silver"><font face="Tahoma" size="2"
color="WHITE"><b>| <a href="incluir-form.asp">Incluir</a>  </b></font></td>
</tr>

</table>

<%
Session("default_ConnectionString") =
"DBQ=c:\inetpub\wwwroot\noticias\noticia.mdb;Driver={Microsoft Access Driver
(*.mdb)}"

Set oRst = Server.CreateObject("ADODB.Recordset")
oRst.Open "SELECT * FROM tnoticia",  Session("default_ConnectionString"),
adOpenKeyset, adLockOptimistic, adCmdText
intQtdRegs = oRst.RecordCount



if intQtdRegs <> 0  then
avarResults = oRst.GetRows()

Response.write "<br><br>"
Response.write "<table align=CENTER>"
            Response.write "<tr>"
            Response.write "<td bgcolor=Navy>"
            Response.write "<font color=White face=Arial size=2><b> Código
</b>"
            Response.write "</td>"
            Response.write "<td bgcolor=Navy>"
            Response.write "<font color=White face=Arial size=2><b> Título da
Noticia </b>"
            Response.write "</td>"
            Response.write "<td bgcolor=Navy>"
            Response.write "<font color=White face=Arial size=2><b> Disponível
</b></font>"
            Response.write "</td>"
```

```
            Response.write "</tr>"

      For i=0 to intQtdRegs
            if (i+1) > intQtdRegs then Exit For
            Response.write "<tr>"
            Response.write "<td bgcolor=#D9D9FF align=center>"
            Response.write avarResults(0,i)
            Response.write "</td>"
            Response.write "<td bgcolor=#D5FFFF>"
            Response.write avarResults(1,i)
            Response.write "</td>"
            Response.write "<td bgcolor=#D9D9FF>"
            Response.write avarResults(3,i)
            Response.write "</td>"
            Response.write "</tr>"
      Next
Response.write "</table>"
Response.write "<br><center><font color=Black face=Arial
size=1><b>Encontrado(s)</b> " & intQtdRegs & " notícia(s)</center>"

Else
Response.write "<br><br><br><center><font color=Black face=Arial size=1><b>Nao
há nenhuma noticia cadastrada!<br>Clique em <b>INCLUIR</b>  para adicionar
noticias </center>"
End if
Set oRst = Nothing
%>

</BODY>
</HTML>
```

O Link Incluir

Esse link chama o script Incluir-form.asp, que contém o formulário onde vai ser digitados o Titulo e o Texto da notícia (Fig. 7).

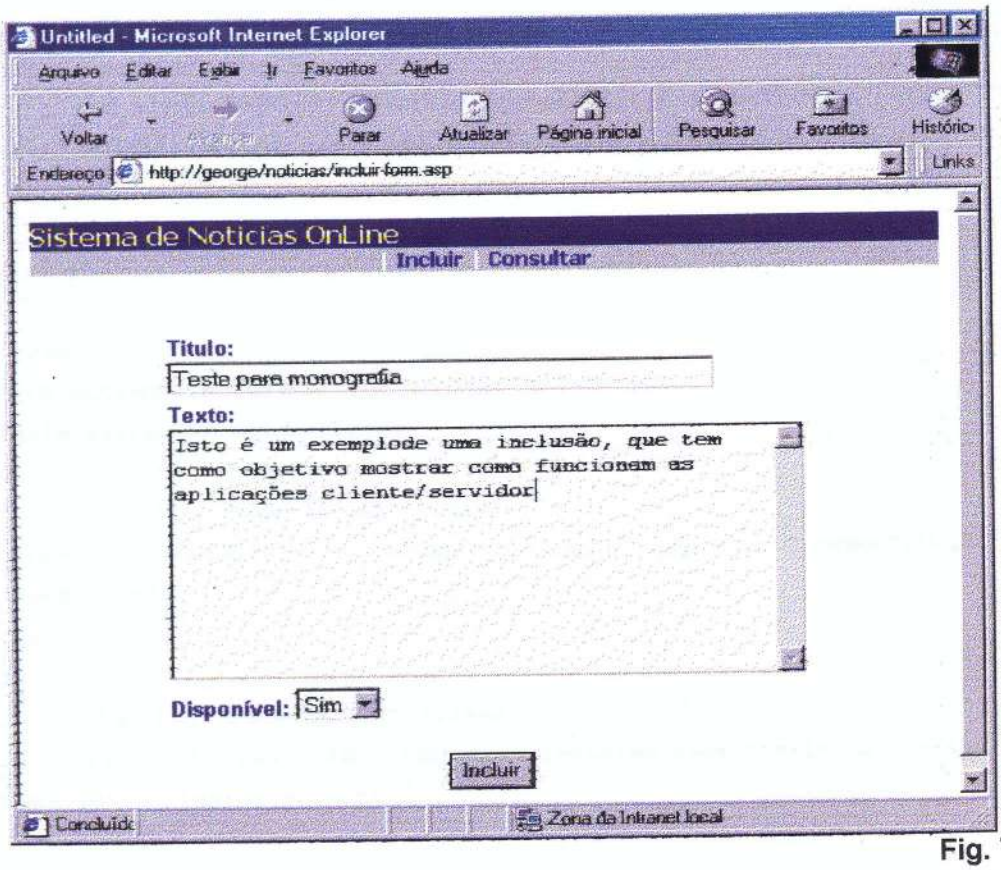

Fig. 7

O código do script incluir-form.asp é

```
<%@ LANGUAGE="VBSCRIPT"%>
<HTML>
<HEAD>
      <TITLE>Untitled</TITLE>
</HEAD>
```

```
<BODY bgcolor="White">
<table width=100% cellspacing="0" cellpadding="0">
<tr>
      <td bgcolor="Navy"><font face="Tahoma" size="3"
color="Yellow"><b>Sistema de Noticias OnLine</b></font></td>
</tr>
<tr>
      <td align="CENTER" bgcolor="Silver"><font face="Tahoma" size="2"
color="WHITE"><b>| <a href="incluir-form.asp">Incluir</a> </b></font></td>
</tr>

</table>

<br><br>
<form action="Incluir-action.asp" method="POST">
<table width=50% align=center>
<tr>
      <td><font face="Arial" size=2
color="Navy"><b>Titulo:</b></font><br><input type="Text" name="Titulo" Value=""
size=45></td>
</tr>
<tr>
      <td><font face="Arial" size=2
color="Navy"><b>Texto:</b></font><br><textarea name="texto" cols="45"
rows="10"> </textarea></td>
</tr>

**<tr>
      <td><font face="Arial" size=2 color="Navy"><b>Disponivel:
</b></font><br><select name='Disponivel'> <option value'Sim>Sim</option>
<option value'Nao'>Não'</option></select></td>
</tr>
</table>
<br>
<center><input type=Submit name=Incluir Value=Incluir></center>
</form>
</BODY>
</HTML>
```

Observe no action do form que os dados digitados nesse formulário será postado no arquivo inclui-action.asp. Que será o responsável por fazer a conexão com a base para incluir os dados. Veja seu código:

incluir-action.asp

```
<%@ LANGUAGE="VBSCRIPT"%>
<HTML>
<HEAD>
      <TITLE>Untitled</TITLE>
</HEAD>
<BODY bgcolor="White">
<table width=100% cellspacing="0" cellpadding="0">
<tr>
      <td bgcolor="Navy"><font face="Tahoma" size="3"
color="Yellow"><b>Sistema de Noticias OnLine</b></font></td>
</tr>
<tr>
      <td align="CENTER" bgcolor="Silver"><font face="Tahoma" size="2"
color="WHITE"><b>| <a href="incluir-form.asp">Incluir</a>   </b></font></td>
</tr>

</table>

<%
Set rstExibe = Server.CreateObject("ADODB.Recordset")
rstExibe.Open "SELECT * FROM tnoticia WHERE CODID=" & -1,
Session("default_ConnectionString"), adOpenKeyset, adLockOptimistic, adCmdText
intQtdRegs = rstExibe.RecordCount
rstExibe.AddNew
rstExibe("titulo_Noticia") = Request.Form("Titulo")
rstExibe("texto_Noticia") = Request.Form("Texto")
rstExibe("disponivel") = Request.Form("Disponivel")
rstExibe.Update
%>
<br><br>
<center><font face="Tahoma" size="3" color="Navy">Noticia <b>INCLUIDA</b> com
sucesso</font></center>
```

```
<center><font face="Tahoma" size="3" color="Navy"><a
href="exibe.asp">VOLTAR</a></font></center>
</BODY>
</HTML>
```

Veja como ficaria, após ter incluído uma noticia, a interpretação do arquivo exibe.asp (Fig. 8)

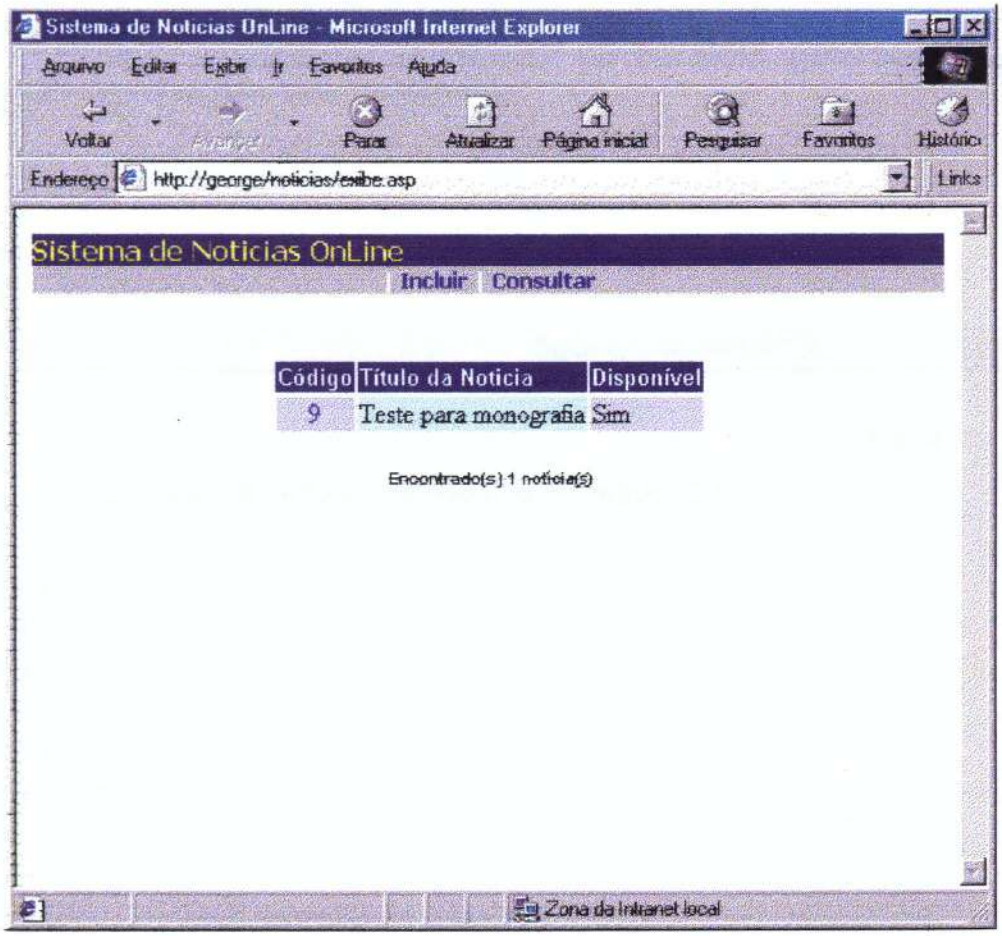

Fig. 8

E agora o arquivo, ver.asp, que exibirá as noticias cadastradas em qualquer site da internet, bastando apenas informar sua localização. Veja sua interpretação pelo browser na figura 9.

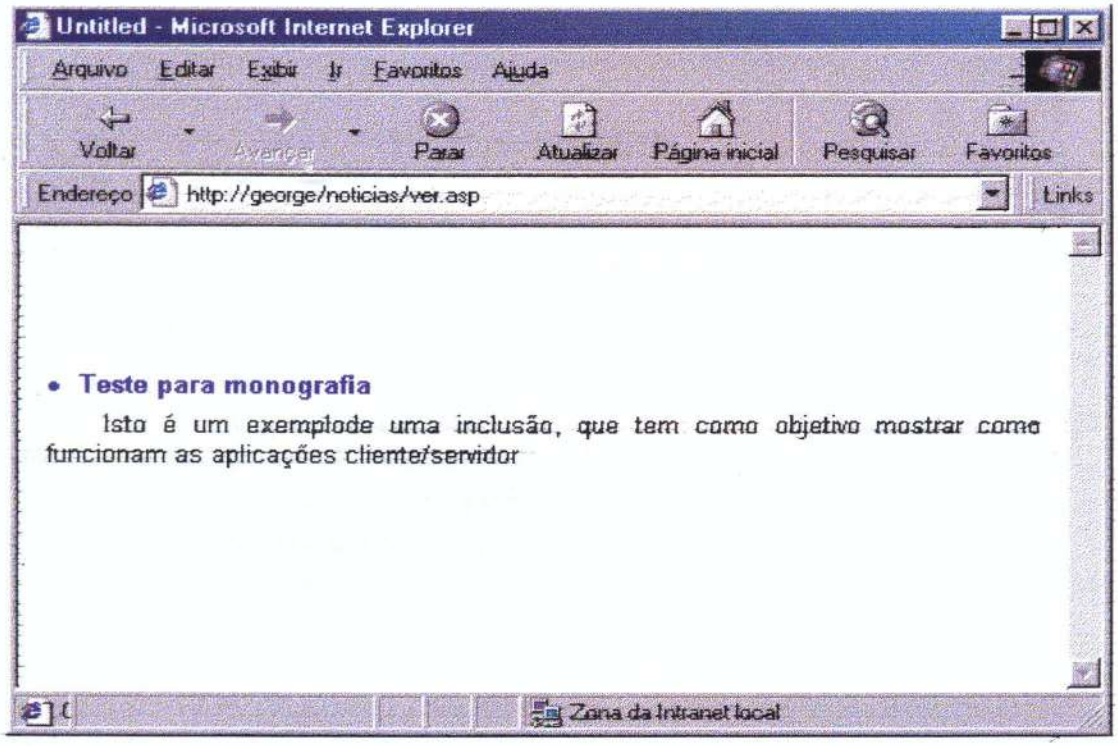

Fig. 9

O código do arquivo ver.asp (Fig. 9) segue abaixo:

```
<%@ LANGUAGE="VBSCRIPT"%>
<HTML>
<HEAD>
      <TITLE>Untitled</TITLE>
</HEAD>
<BODY bgcolor="White">

<%
Session("default_ConnectionString") =
"DBQ=c:\inetpub\wwwroot\noticias\noticia.mdb;Driver={Microsoft Access Driver
(*.mdb)}"

Set oRst = Server.CreateObject("ADODB.Recordset")
oRst.Open "SELECT * FROM tnoticia WHERE Disponivel='Sim' ORDER By CODID DESC" ,
Session("default_ConnectionString"), adOpenKeyset, adLockOptimistic, adCmdText
intQtdRegs = oRst.RecordCount

if intQtdRegs <> 0  then
```

```
avarResults = oRst.GetRows()

Response.write "<br><br>"
Response.write "<table width=500 align=CENTER>"

      For i=0 to intQtdRegs
            if (i+1) > intQtdRegs then Exit For
            Response.write "<tr>"
            Response.write "<td>"
            Response.write  "<br>"
            Response.write "<font color=Blue size=2 face=Arial><li><b>" &
avarResults(1,i) & "</b></font>"
            Response.write "</td>"
            Response.write "</tr>"
            Response.write "<tr>"
            Response.write "<td>"
            Response.write "<p align=JUSTIFY><font color=Black size=2
face=Arial>       " & avarResults(2,i) &
"</font>"
            Response.write "</td>"
            Response.write "</tr>"
      Next
Response.write "</table>"


Else
Response.write "<br><br><br><center><font color=Black face=Arial size=1><b>Nao
há nenhuma noticia cadastrada!</center>"
End if
Set oRst = Nothing
%>

</BODY>
</HTML>
```

11 Conclusão

As ferramentas de desenvolvimento apresentados estão cada vez mais atraindo adeptos para o desenvolvimento de aplicações para internet que possui uma grande gama de aplicações e um futuro muito promissor.

Os ASP, na minha opinião, é a ferramenta que melhor atende as necessidades no desenvolvimento de um projeto. Para os programadores em Visual Basic fica mais fácil migrar para o ASP, já que o código é bem parecido com o Visual Basic. Sua performance em acesso a banco de dados via ADO é excelente o que é muito importante quando se trabalha com aplicações Cliente/Servidor.

O Cold Fusion é a ferramenta mais fácil de se trabalhar com aplicações Cliente/Servidor já que seus comandos são bem parecidos com as tags html, embora seja a que apresenta menos recursos. Seu acesso a banco de dados, feito por um DataSource criado em seu Servidor, apresenta um resultado inferior no quesito de performace, contrário a facilidade que apresenta o DataSource na hora de se fazer uma consulta, inclusão, alteração ou exclusão. A Allaire, empresa que desenvolveu o Cold Fusion, já o lançou para o Ambiente Unix/Linux.

Perl é uma boa solução para o desenvolvimento de aplicações internet no ambiente Unix/Linux. Com certeza é um pouco mais complicado de se usar do que o ASP ou ColdFusion já que é bem parecido com a linguagem C. O Perl encontra concorrentes fortes como o PHP, para ambiente Unix/Linux, que apresenta algumas facilidades não encontradas no Perl, como por exemplo no tratamento dos campos postados de um formulário que ao invés de retirar os valores dos campos da QUERYSTRING, eles podem ser recebidos diretamente como no ASP. Outra concorrente é a Python, bem parecida com Perl, apresenta um sintaxe fácil alem de ser orientada a objeto o que a torna mais avançada em termos de eficiência e extensibilidade. A Python é uma linguagem relativamente nova que merece atenção.

ASP, COLD FUSION e PERL são com certeza as ferramentas mais populares da internet, espero que este projeto tenha passado um pouco sobre o funcionamento

de cada uma. E se você pretende entrar no campo de desenvolvimento de aplicações para internet, escolha aquela ferramenta que mais atende a sua necessidade.

Bibliografia

HOOZER, L.M.V. (1997) Apreenda em 21 dias Microsoft Visual Interdev. 1º ed. CAMPUS.

LARRY W., TOM C., RANDAL L.., STEPHEN P. (1996) Programming Perl. 2º ed. O'Reilly.

WWW (World Wide Web)

ASPBRASIL. Site voltado aos desenvolvedores em ASP com exemplos, fóruns e artigos. (12 mar.).
http://www.aspbrasil.com.br

APS 101. Site voltado aos desenvolvedores em ASP, com exemplos, dicas e artigos. (10 abr.).
http://www.asp101.com

PERL. Site com guias de referência e documentação sobre Perl. (20 abr.).
http://www.perl.com

COLD FUSION BRASIL. Site voltado aos desenvolvedores em Cold Fusion com documentação, fóruns, exemplos e artigos. (12 mar).
http://www.coldfusionbrasil.com.br

ALLAIRE COORPORATION. Site da empresa que desenvolveu o Cold Fusion, com informações e manuais.
http://www.coldfusion.com