ASP - Active Server Pages

O PROBLEMA

Introdução

Anos 60, auge da guerra fria, o governo americano através de seu Departamento de Defesa e da recém criada ARPA (Agência de Projetos de Pesquisa Avançada), investe na pesquisa e desenvolvimento de uma rede de comunicação que pudesse sobreviver a um possível ataque nuclear, onde a perda de uma parte da rede, não comprometesse o funcionamento das demais partes.

Fruto dessa pesquisa, em setembro de 1969 surgiu o primeiro nó da então chamada ARPAnet, pela ligação do primeiro servidor na UCLA (University of California at Los Angeles). Posteriormente, no mesmo ano, outros três nós foram instalado no SRI (Stanford Research Institute), na UC Santa Barbara e a Universidade de Utah.

Nos anos seguintes, foi crescente o aumento das instituições acadêmicas envolvidas no aperfeiçoamento e desenvolvimento dessa nova tecnologia, bem como, no seu emprego em pesquisas e compartilhamento de informações. Em janeiro de 1983 entrou em cena o TCP/IP, protocolo capaz de interligar redes distintas, que substituiu o NCP e impulsionou ainda mais o crescimento da rede.

Nos anos 80, dezenas de vendedores incorporaram TCP/IP em seus produtos porque viram compradores para aquele modelo de rede. Era o início da comercialização da Internet. Envolvendo não somente o desenvolvimento de serviços privados e competitivos mas também produtos comerciais implementando a tecnologia Internet.

Hoje vivemos numa realidade de fragmentos de conhecimento. Os indivíduos controlam as ações de partes e não mais do todo. Foco em conhecimento pressupõe a preocupação com a eficiência financeira, melhor performance, o objetivo de se tornar líder de mercado, o fazer mais com menos, e o ajuste a contingências quaisquer. Conhecimento não é igual a informação. O conhecimento e o valor construído diariamente quando o focalizamos, é igual à análise e à ação em cima da informação.

A Internet, como rede mundial de computadores interconectados, é um privilégio da vida moderna para o homem moderno. É o maior repositório de informações acessíveis a qualquer pessoa que a acesse de qualquer parte do mundo. E o que torna a Internet tão diferente das outras invenções humanas é o insignificante período de tempo em que ela precisou para ser usada por milhões de pessoas. A eletricidade (1873), por exemplo, atingiu 50 milhões de usuários depois de 46 anos de existência. O telefone (1876) levou 35 anos para atingir esta mesma marca. O automóvel (1886), 55 anos. O rádio (1906), 22 anos. A televisão (1926), 26 anos. O forno de microondas (1953), 30 anos. O microcomputador (1975), 16 anos. O celular (1983), 13 anos. A Internet (1995), por sua vez, levou apenas 4 anos para atingir 50 milhões de usuários no mundo. Hoje já temos 221 milhões de pessoas acessando a Internet. 57% dos internautas de hoje tem inglês como idioma nativo. 43% falam outros idiomas.

O maior propulsor para este crescimento vertiginoso foi a introdução, em 1990, de um

protocolo chamado HTTP (hypertext transmission protocol), da linguagem HTML (hypertext markup language) e do Web Browser (aplicativo que interpreta o código HTML e exibe as páginas) o que permitiu o desenvolvimento de aplicações gráficas, e o surgimento da WWW (World Wide Web).

O HTML instrui o Browser como exibir a informação contida num documento através de uma formatação padronizada que será interpretada. O Hipertexto é uma tecnologia que permite a ligação entre documentos ou páginas relacionadas através de links. Quando o usuário ativa (clica) um link, a página apontada pelo link é exibida. O interessante é que a pagina apontada pelo link pode se encontrar em qualquer servidor, de qualquer parte do mundo, criando uma verdadeira teia. Páginas Web são simplesmente arquivos texto que seguem uma formatação padronizada pelo HTML e armazenadas em servidores. Cada servidor tem seu próprio endereço na Internet. O endereço do servidor, o caminho do diretório e o nome do arquivo são combinados para formar a URL (Uniform Resource Locator), o endereço da página.

Basicamente, temos dois elementos atuando na Internet: o cliente que requisita uma série de informações da rede e um servidor que responde a esta requisição fornecendo a informação solicitada. Quando o usuário se conecta a Internet e acessa uma página ou envia um e-mail, ele está solicitando um serviço à rede. O servidor recebe o pedido e providencia a resposta à sua solicitação.

É interessante notar que o cliente não precisa saber nada sobre o computador que faz o papel de servidor, nem o servidor precisa saber nada sobre o computador que faz o papel do cliente. O protocolo de comunicação é o responsável pela interação.

Os protocolos mais utilizados na Internet são o HTTP e o FTP (File Transfer Protocol) protocolo responsável pela transferência de arquivos. É através destes protocolos que a comunicação do cliente com o servidor é feita na Web.

No início, as páginas de documentos localizadas nos servidores, apesar de poderem possuir elementos gráficos, eram estáticas, com atualizações de conteúdo sendo feitas a cada semana ou mês. Porém, com o aumento da popularidade da Internet e da informação disponibilizada, a necessidade de atualização destas páginas, em muitos casos, passou a ser diária e até mesmo instantânea.

Com o incremento dos negócios na internet e todo o seu crescimento comercial, os dados que necessitam ser alterados pelos usuários durante o acesso a uma página tornaram-se uma coisa comum e página que alteram o seu conteúdo a cada interação com o usuário passaram a ser mais frequentes. Desta forma surgiu o conceito de Páginas Dinâmicas.

Formulação da Situação-Problema

Páginas Dinâmicas foram inicialmente criadas com o uso de uma tecnologia chamada CGI (Common Gateway Interface), onde programas executados no servidor respondem às solicitações dos usuários para criar páginas dinamicamente, personalizar páginas e manipular dados.

O CGI, apesar de ser uma tecnologia segura e confiável, é relativamente difícil de implementar, sendo geralmente escrito em linguagem "C" ou "Perl". Programas CGI

também consomem mais recursos (memória) do servidor isto porque a cada solicitação do cliente é criado um novo processo, ou seja, todo o código é novamente executado, e, justamente pelo fato dos programas CGI serem independentes - não havendo um depósito global de informações relativas àquele usuário -, eles não são capazes de manter a persistência dos dados memorizando seu estado (variáveis) durante a transição de páginas pelo usuário o que é muito importante em alguns casos.

Uma evolução desta tecnologia para acesso e atualização dinâmica proposta pela Microsoft é o ASP (Active Server Pages). Com o uso de linguagens de alto nível e a capacidade de criar automaticamente um ambiente para cada usuário, através dele podemos criar páginas interativas de forma rápida e fácil o que vem tornado-o cada vez mais popular.

A essência do problema é, portanto, analisar os motivos que tornam o ASP uma ferramenta cada vez mais utilizada, sua aplicabilidade, restrições, limitações e facilidades na geração de páginas dinâmicas.

Objetivo do Presente Trabalho

O presente estudo busca apresentar a aplicação do ASP na Internet e seu funcionamento, características e estrutura.

Questões de Estudo

1 Quando ocorre a percepção da necessidade do uso de Páginas dinâmicas ?
2 Porque usar o ASP ?
3 O ASP suporta banco de dados ?
4 Que ambiente preciso pra executar as páginas ASP ?

A Importância do Estudo

A importância deste estudo se dá tendo em vista o crescente aumento de situações onde o emprego de páginas dinâmicas e acesso a banco de dados são necessários.

Outra variável a ser considerada por nós, profissionais de informática, é o grande campo de trabalho que esta especialização representa, pois para a extração de todo o potencial desta nova tecnologia, a ação de um profissional com o devido conhecimento se torna item imprescindível.

Delimitação

O estudo fica limitado à apresentação do ASP como ferramenta importante na busca da interatividade e personalização na Internet e a linguagem de scripts utilizada nos

exemplos será o VBScript.

CAPÍTULO II

DESENVOLVIMENTO

A Internet a cada dia que passa se torna mais dinâmica. O comércio eletrônico, a publicação de jornais e pesquisas e a personalização de produtos e serviços são atividades cada vez mais constantes exigindo interfaces e aplicações cada vez mais específicas inerentes ao seu público alvo ou área de atuação. Isto se deve principalmente à concorrência entre empresas na busca do melhor atendimento às necessidades a seus clientes, na ampliação e especialização de seu nicho de mercado e na redução de custos operacionais.

O desenvolvimento de páginas HTML é uma solução fácil, rápida e muito utilizada, porém, sua aplicação nesses casos é limitada. O conteúdo dessas páginas após ser escrito precisará ser formatado em HTML para então ser publicado. Ao ser publicado, as páginas com os links que irão fazer referência à essa página precisarão ser atualizadas ou criadas. Esta abordagem é a ideal para sites onde alterações de conteúdo não seja constante, não precisem reter informações sobre seus visitantes ou clientes para fins de personalização, não exijam proteções por senha nem disponibilizem bancos de dados.

A necessidade de um sistema mais elaborado fica clara à medida que essas características, ou mesmo parte delas, se tornam indispensáveis. Nesses casos o emprego de programas auxiliares para dar o suporte necessário é a única saída. Entram em cena então os Scripts. Scripts são códigos que podem ser escritos em diversas linguagens de programação tais como "C", "C++", "Perl", "JavaScript", "VBScript" entre outras, estes scripts são executados no servidor web (host) a medida que são requisitados pelas páginas web para executar determinada ação para o qual foi projetado, por exemplo: manipular bancos de dados, validar senha de usuários, executar cálculos, enviar e-mails, publicar dados e, é claro, personalizar páginas, entre tantas outras. Após a execução no servidor, somente as respostas (HTML) são enviadas ao cliente, garantido a proteção do código.

Vejamos como exemplo um jornal que publique seu conteúdo na Internet. Imagine a quantidade de links que haverão para acesso às diversas matérias publicadas, pense ainda que as notícias deverão ser publicadas o mais rápido possível e que cada notícia será uma página HTML. Nesse caso, a notícia seria editada e formatada para o HTML, enviada ao servidor, e todas as páginas que farão referência a esta notícia precisarão ser atualizadas. Imagine, ainda, isso numa redação de jornal onde chegam centenas de notícias por dia. É simplesmente impraticável.

A solução para este problema está na geração dinâmica de páginas, ou seja, sob demanda, onde páginas modelo são alimentadas por um complexo sistema de banco de dados que armazena somente o conteúdo de cada notícia com suas características e classificação. Essas páginas, após mesclagem, são então enviadas ao browser do usuário. Veja abaixo um exemplo simplificado.

Atualmente no mercado existem diversas ferramentas capazes de gerar páginas

dinâmicas, porém o ASP vem se destacando rapidamente das demais devido às suas características. Mas, afinal, o que é o ASP ?

ASP é a abreviação de Active Server Pages ou Servidor de Páginas Ativas. É um ambiente desenvolvido pela Microsoft para uma eficiente codificação de scripts implementados para a execução em servidores web em resposta à solicitações de usuários. Neste ambiente é possível combinar HTML, scripts, bancos de dados e componentes Activex reutilizáveis para a criação de poderosas soluções comerciais para a Web.

O ASP é multi-thread e multi-usuário, oferece suporte nativo ao VBScript e JScript, suporta controles ActiveX, bancos de dados ODBC e servidores SQL. Detalhemos melhor isso...

Em primeiro lugar devemos observar possíveis linguagens empregadas na implementação de seus scripts. O ASP oferece suporte nativo ao VBScript e ao JScript. Porém, é capaz de suportar praticamente qualquer linguagem de script através da adoção de módulos Activex adicionais, por exemplo o plug-in para Perlscript do fabricante ActiveWare.

O VBScript é uma linguagem de scripts de sintaxe e estrutura muito similar ao Visual Basic usado no desenvolvimento de aplicações comerciais. Trata-se de uma linguagem de alto nível, de fácil aprendizado e muito utilizada no mercado, podendo-se afirmar que quem sabe programar com Visual Basic não terá dificuldades com VBScript, este é realmente um dos seus grandes atrativos. Por outro lado, quem já domina alguma outra linguagem, como por exemplo JavaScript ou PerlScript entre outras, não é obrigado a aprender VBScrpipt, uma vez que o ASP é capaz de prover suporte a estas linguagens também.

Outra característica que chama a atenção é a facilidade de desenvolvimento de aplicações que se utilizem de banco de dados. O ASP oferece suporte total ao padrão ODBC e servidores SQL, o que significa capacidade de acessar mais de 55 tipos diferentes de bancos de dados. Através do controle Activex Data Objects (ADO) são oferecidos métodos e propriedades capazes de tornar fácil, muito rápido e transparente o acesso e a manipulação de dados.

Uma grande dor de cabeça para muitos desenvolvedores, conforme foi apresentado na Formulação da Situação-Problema deste trabalho, é a necessidade da manutenção de estado, isto é, para cada cliente (usuário) armazenar o valor de variáveis enquanto ele navega entre as páginas. A solução proposta pelo ASP é o uso do objeto Session. Um novo objeto Session é criado para cada usuário que acessa a aplicação e, dentro dele, podemos criar variáveis e funções que ficarão disponíveis somente para aquele usuário durante sua navegação pela aplicação podendo ser acessados ou alterados a qualquer momento, além do objeto Session, temos, também o objeto Application porém este é "visível" a todos os usuários. Vejamos o exemplo:

As limitações de performance impostas pela execução de um novo processo a cada requisição, acarretando consumo maior de memória e processamento, foram superadas. O ASP foi otimizado para suportar múltiplos usuários e múltiplas threads, o que torna seu processamento muito mais rápido. Para tanto, a execução de Scripts é tratada como um serviço, não como um processo, economizando memória e aumentando performance.

Além dessas características relevantes, o ASP suporta componentes ActiveX escritos em praticamente qualquer linguagem, incluindo Java, Visual Basic, C++, entre outras. O emprego de componentes ActiveX aumenta significativamente o poder e a simplicidade da aplicação desenvolvida.

Outra facilidade é a forma como podemos mesclar o código do script com o código html. Uma página ASP nada mais é que um arquivo texto com extensão ".asp". A codificação do script deverá estar delimitada entre as seguintes tags: "**<%**" e "**%>**". Tudo que estiver entre estas tags será processado no servidor e o que estiver fora, ou seja, o código HTML, será enviado normalmente ao cliente que requisitou a página. Vejamos o exemplo:

Este exemplo verifica a hora atual do servidor e envia ao browser do usuário uma das opções, dependendo da hora. Note em negrito os comandos do VBScript entre as tags de delimitação do ASP, o resto todo é HTML puro, observe também a flexibilidade no emprego das tags ASP e sua mesclagem ao código HTML.

Para podermos efetivamente desenvolver páginas ASP, precisamos conhecer um pouco de sua estrutura de objetos internos.

A Estrutura de Objetos ASP

O ASP possui cinco objetos padrão, são eles:

REQUEST – Para retornar ao servidor informações do usuário.

RESPONSE – Para enviar informações ao usuário.

SERVER – Para Controlar o Internet Information Server.

SESSION – Para armazenar informações do usuário da sessão corrente.

APPLICATION – Para compartilhar informações entre todos os usuários .

Abaixo temos a hierarquia e detalhamento de cada um dos objetos:

- ***O Objeto Request***

O Objeto Request retorna as informações contidas nas requisições feitas pelo Browser do cliente ao servidor.

Cada requisição pode ter diversos parâmetros em diferentes categorias e, para tratar esta quantidade de informação o objeto Request utiliza-se de coleções (Collections) que

podem ser vistas como um vetor contendo um conjunto de informações. Suas coleções são:

- **Form** – Para obter informações de um Formulário HTML quando o Method usado no Form for o POST.

  Sintaxe: Request.Form(parametro)[index][.count]

- **QueryString** – Para obter os parâmetros anexados ao fim do endereço URL da página, ou quando o Method usado no Form for o GET.

  Sintaxe: Request.QueryString(variavel)[(index)][.count]

- **ServerVariables** – Contém toda a informação gerada quando da requisição de serviço pelo Browser combinada com as variáveis do ambiente do servidor.

  Sintaxe: Request.ServerVariables(variável)

- **Cookies** – Esta coleção permite tratar os valores dos cookies enviados em uma requisição HTTP. Cookies são arquivos no formato txt gravados na máquina do usuário com múltiplas finalidades, por exemplo: guardar preferências dele para uso em uma próxima visita ao site.

  Sintaxe: Request.Cookies(cookie)[(chave)][.atributo]

- **ClientCertificate** – Refere-se ao uso de certificados pelos usuários quando do acesso a um site seguro como meio de identificação.

  Sintaxe: Request.ClientCertificate(chave[subcampo])

- ***O Objeto Response***

  Este objeto é usado para enviar informações ao cliente. Possuindo apenas uma coleção, a Cookies, este objeto possui também um conjunto de cinco propriedades e sete métodos.

- **Cookies** – Esta coleção determina o valor de um cookie. Se o cookie especificado não existir, ele será criado no cliente, se já existir, será atualizado com o novo valor.

  Sintaxe: Response.Cookies(Cookie)[(chave)][.atributo]

Propriedades

- **Expires** – Esta propriedade representa o tempo em minutos antes do conteúdo da página se expirar, ou seja, determina o tempo durante o qual o browser usará a página do cache, ao expirar este tempo a página será solicitada novamente ao

servidor, garantindo o acesso às informações mais atualizadas.

Sintaxe: Response.Expires [=valor]

- **ExpiresAbsolute** – Idêntica a Expires porém definindo uma data e hora.

Sintaxe: Response.ExpiresAbsolute [=(data)(hora)]

- **ContentType** – Muda o cabeçalho HTTP da página para indicar que tipo de dados a página contém. Seu padrão é "text/HTML".

Sintaxe: Response.ContentType [=tipo]

- **Status** – Permite definir o status da linha HTTP que é devolvida ao Browser, consistindo em um código de 3 dígitos.

Sintaxe: Response.Status =descricaodostatus

- **Buffer** – Determina se o conteúdo da página gerada pelo Script será enviada ao Browser conforme execução ou se tudo é enviado no final da execução. False – sem buffer (padrão). True – com buffer.

Sintaxe: Response.Buffer [=valorlógico]

Métodos

- **AddHeader** – Inclui um cabeçalho HTML com o valor especificado.

Sintaxe: Response.AddHeader nome, valor

- **AppendToLog** – Inclui uma string ao fim da entrada do log do servidor para esta requisição.

Sintaxe: Response.AppendToLog string

- **Clear** – Remove o conteúdo de qualquer página do buffer quando o buffer estiver ativado.

Sintaxe: Response.Clear

- **End** - Encerra o processamento do script atual.

Sintaxe: Response.End

- **Flush** – Usado para enviar o conteúdo do buffer imediatamente ao cliente quando o buffer estiver ativado.

Sintaxe: Response.Flush

- **Redirect** – Redireciona o cliente para uma outra página ou endereço.

Sintaxe: Response.Redirect enderecodestino

- **Write** – Permite escrever textos na página gerada.

Sintaxe: Response.write texto

O Objeto Server

Este objeto fornece métodos e propriedades que permitem interagir com o servidor que hospeda a aplicação ASP através de uma propriedade e quatro métodos como veremos:

Propriedades

- **ScriptTimeout –** Especifica quantidade máxima de tempo em segundos que um script pode executar até terminar.

Sintaxe: Server.ScriptTimeout =segundos

Métodos

- **CreateObject** – Este objeto é, provavelmente, o mais importante dos objetos ASP internos. Ele cria uma instância de um componente (ActiveX, por exemplo) no servidor. Após criar um componente podemos então utilizar suas propriedades e métodos. O escopo do objeto criado pode variar desde o nível de aplicação, sessão ou script atual.

Sintaxe: Server.CreateObject (tipodeobjeto)

- **HTMLencode** – Converte os caracteres especiais que não podem ser escritos diretamente nas páginas HTML.

Sintaxe: Server.HtmlEncode (string)

- **MapPath** – Retorna o caminho físico da localização do arquivo no servidor.

Sintaxe: Server.MapPath (caminho)

- **URLencode** – Converte os caracteres não permitidos usados em um endereço URL em caracteres permitidos.

  Sintaxe: Server.UrlEncode (string)

O Objeto Application

Este objeto pode ser utilizado para compartilhar informações entre todos os usuários de uma aplicação. Uma aplicação ASP é definida como sendo todos os arquivos .asp dentro do diretório virtual e seus subdiretórios. É criado quando a aplicação é publicada no servidor e persiste para todos os usuários por toda a vida da aplicação. Neste objeto, podemos criar variáveis globais visíveis a todos os clientes. Possui somente dois métodos e também dois eventos relacionados. Vejamos:

Métodos

- **Lock** e **UnLock** – Permite ou Não que outros clientes modifiques as propriedades do objeto application, respectivamente.

  Sintaxe: Application.Lock

Eventos

- **Application_OnStart** e **Application_OnEnd** – Eventos executados quando a aplicação é iniciada ou encerrada, respectivamente.

O Objeto Session

Este objeto é semelhante ao Application, entretanto, ao contrário do objeto Application armazena informações de uma sessão de um usuário em particular. O servidor cria automaticamente um objeto Session quando uma página asp é requisitada pelo usuário que ainda não tenha uma sessão. Persistindo para a sessão inteira, o objeto é destruído quando a sessão for abandonada ou quando expirar. Nele podemos definir variáveis globais para cada cliente, por exemplo, gerenciar o acesso com senha. Através deste objeto temos a solução para o problema da persistência de estado. Para podermos usar o potencial deste objeto, o Browser do usuário deve ser capaz de suportar cookies e esse suporte deve estar habilitado. Vejamos suas propriedades e eventos:

Propriedades

- **SessionID** – Identifica especificamente uma sessão e é única em toda a instância da aplicação.

  Sintaxe: Session.SessionID

- **Timeout** – Define ou retorna a quantidade de minutos antes da sessão ativa expirar.

  Sintaxe: Session.Timeout [=minutos]

- **Abandon** – Encerra a sessão atual imediatamente.

  Sintaxe: Session.Abandon

Eventos

- **Session_OnStart** e **Session_OnEnd** - Eventos executados quando uma sessão é iniciada e encerrada, respectivamente. Pode ser usada para setar valores iniciais de variáveis ou instanciar objetos, por exemplo.

O Arquivo Global.asa

O arquivo Global.asa é um arquivo texto que contém declarações gerais com escopo ao nível da aplicação. Existindo somente um para cada aplicação, é neste arquivo que definimos os eventos Application_OnStart, Application_OnEnd, Session_OnStart e Session_OnEnd. Veja abaixo um exemplo vazio:

```
<SCRIPT LANGUAGE=VBSCRIPT RUNAT=SERVER>

    Sub Application_OnStart '' EndSub

    Sub Application_OnEnd '' EndSub

    Sub Session_OnStart '' EndSub

    Sub Session_OnEnd '' EndSub

</SCRIPT>
```

Veja a seguir um exemplo de implementação do global.asa para um contador de usuários ativos:

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

    Sub Application_OnStart

    'Zeramos o contador, uma variável a nível de aplicação.

    Application("contador") = 0

    EndSub

    Sub Application_OnEnd '' EndSub

    Sub Session_OnStart

    'Altera o timeout da sessão para 20 minutos

Session.Timeout = 20

'Incrementa o contador cada vez que a sessão é iniciada.

Application.Lock

Application("contador") = Application("contador") + 1

Application.UnLock

EndSub

Sub Session_OnEnd

'Decrementa o contador quando cada sessão é encerrada.

Application.Lock

Application("contador") = Application("contador") - 1

Application.UnLock
```

```
EndSub

</SCRIPT>
```

O resultado poderá ser exibido em qualquer página pelo script a seguir:

```
<%@ Language=VBScript %>

<HTML>

<HEAD>

<TITLE> Mostra Usuários </TITLE>

<BODY>

<B><FONT COLOR="#CC0000">

Número de Usuários Ativos: <%=Application("contador")%>

</FONT></B>

</BODY>

</HTML>
```

Os Tipos de Dados e variáveis

Os tipos de dados utilizados nos scripts, bem como a definição de variáveis e estrutura de programação, são pertinentes à linguagem utilizada. Nos scripts aqui demonstrados é utilizado o VBScript.

Criação de Arquivos e Funções Globais

Um dos recursos do ASP é o *include* no lado do servidor. Este recurso permite que você inclua arquivos arbitrários dentro de uma página ASP durante a execução. Isto é extremamente útil para a criação de funções globais, cabeçalhos, rodapés ou outros

elementos que precisem ser reutilizados em várias páginas.

Dessa forma, se houver necessidade alterar essas funções ou fragmentos HTML, poderá fazê-lo uma vez só e a alteração será refletida automaticamente em todas as paginas que fazem referência a este arquivo.

Seu único inconveniente é a inserção de uma pequena sobrecarga no servidor em função da necessidade de localização do arquivo e da mesclagem dos códigos.

Sintaxe: <!-- #INCLUDE VIRTUAL|FILE = "nomedoarquivo" -->

Ambiente de servidor necessário ao ASP

O ASP exige para sua execução que a plataforma utilizada seja o Windows NT porém, o Windows NT Workstation, o Windows 95 ou o Windows 98 podem ser usados para desenvolvimento dos scripts. Uma empresa chamada ChiliSoft's desenvolveu um produto capaz de executar o ASP em outras plataformas como o Unix e suas variações, por exemplo.

É necessário, também, a presença do servidor Web Microsoft Internet Information Server (IIS) ou o Personal Web Server (PWS) instalado. Ambos já trazem embutido o suporte ao ASP, às linguagens VBScript e Jscript e componente (ADO) para acesso a banco de dados.

Para acesso a banco de dados, o driver ODBC para o banco utilizado deverá estar, também, instalado no servidor.

Ambiente do cliente necessário ao ASP

Para os clientes não há exigências ou restrições quanto ao Browser utilizado, exceto para a necessidade do uso de cookies uma vez que são escritos na máquina do usuário, o browser deve oferecer tal suporte e o mesmo deve estar habilitado. Para o uso do objeto Session, os cookies também devem estar habilitados na máquina do usuário.

Acessando Banco de Dados e Utilizando o Objeto Sessão

Para enviar informações ao ASP utilizamos um formulário HTML ou as passamos como parâmetros no final da URL do script que irá processá-los.

Para isso, no caso do formulário, utilizamos a propriedade Method da Tag Form que deve ser preenchida com "POST" ou "GET". Se utilizarmos "POST" as informações (parâmetros) estarão disponíveis através da coleção Form do objeto Request. Se utilizarmos o "GET", estarão disponíveis no script através da coleção QueryString do objeto Request. Na propriedade Action da Tag Form informamos o script que irá processar as informações daquele formulário, no exemplo abaixo "validar.asp".

No caso de anexarmos os parâmetros à URL, por exemplo: "www.dominio.com/validar.asp?t_login="Roberto"&t_senha="012345", estas informações, assim como no caso do "GET" estarão disponíveis via coleção Request.QueryString.

O script a seguir é de uma página de Login usada para, por exemplo, autorizar o acesso de determinado usuário à páginas protegidas. Neste exemplo, o usuário preenche um formulário (login.asp) informando seu nome de Login e sua Senha. Ao clicar em "Enviar", um script ASP (validar.asp) é acionado para validar as informações preenchidas e, de acordo com a resposta, redirecioná-lo para o ambiente protegido. O objetivo desse script é criar uma variável no objeto sessão, indicando que aquele usuário tem autorização de acesso, no caso, a variável "logedin". Para realmente proteger as páginas que se deseja, no início da página protegida (ex: sejabemvindo.asp) deverá haver um script que teste a existência dessa variável (logedin) e se a mesma não está vazia. Se ela não existir, redirecionamos o cliente para a página de Login e encerramos o processamento (Response.End).

Observe que apesar de ser um arquivo .asp, o seu conteúdo é HTML puro. Veja também, em negrito, o que realmente importará para o ASP.

Abaixo temos a estrutura do arquivo que irá validar o acesso. Para tanto, consideramos o uso de um banco de dados access de nome **"fiaadb"** com uma tabela chamada **"senhas"**.

A página que se deseja proteger deverá ter possuir um script que teste se o usuário tem permissão de acessá-la, como no exemplo abaixo.

O Emprego de Cookies na Personalização de Páginas

Os Cookies permitem que páginas e sites armazenem informações importantes na máquina do cliente. Frequentemente é desejável salvar os dados na máquina do cliente para que estejam disponíveis na próxima vez que o cliente se acessar o site. O maior emprego desta tecnologia se dá em sites que explorem a personalização de suas páginas.

É importante lembrar seu uso fica restrito a Browsers que ofereçam este suporte e que este esteja habilitado.

Os Cookies são escritos na máquina do cliente através da utilização do objeto Response. A coleção Cookies do objeto Request, por outro lado, permite que você leia os Cookies da máquina do cliente e baseado nessas informações, redirecione o cliente para uma determinada página, por exemplo.

Criando Cookies

Neste exemplo, supomos que o usuário está preenchendo um formulário onde o Action da Tag Form é "criacookie.asp" e o Method usado é "GET" para enviar seu nome e a data de nascimento. Criaremos então um Cookie chamado "meucookie" contendo estas informações enviadas, além da data de criação, data de expiração, e domínio.

Como exemplo do emprego, o código abaixo verifica se o mês constante na data de nascimento gravada no Cookie é o mês atual, se for, o usuário será redirecionado para uma página parabenizando usuário pelo aniversário. Este código deverá estar no início da página principal de um site frequentado pelo cliente, por exemplo.

Definição de Termos

Internet

Se refere ao sistema de informação global que -- (i) é logicamente ligado por um endereço único global baseado no Internet Protocol (IP) ou suas subsequentes extensões; (ii) é capaz de suportar comunicações usando o Transmission Control Protocol/Internet Protocol (TCP/IP) ou suas subsequentes extensões e/ou outros protocolos compatíveis ao IP; e (iii) provê, usa ou torna acessível, tanto publicamente como privadamente, serviços de mais alto nível produzidos na infra-estrutura descrita.

Páginas Dinâmicas

Páginas geradas sob demanda cujo conteúdo pode ser personalizado.

CGI

CGI significa "Common Gateway Interface". Protocolo com especificações para transferência de informações entre um servidor web e um programa CGI. Um programa CGI é qualquer programa capaz de aceitar e retornar informações conforme esse protocolo.

Protocolo

Conjunto de regras com finalidade de uniformizar determinado procedimento.

Controles ActiveX

Padrão proposto pela Microsoft para o desenvolvimento de objetos e controles auto-

suficientes no desempenho de determinada função. Controles Activex são objetos compilados específicos para determinada função, por exemplo, fazer interface com um banco de dados.