

# DESENVOLVENDO WEB SITES

scripts, ferramentas, conceitos e dicas

**Projeto de Sites WEB**

**Sites de 3a. Geração**

**Gráficos para WEB**

**IDC - Internet Database Connector**

**Active Server Pages - ASP**

## PROJETO DE SITES WEB

Criar sites web é uma nova área de estudo. Ela deriva de disciplinas tão distintas quanto a Engenharia de Software e o Marketing. A seguir enumero alguns princípios de gerenciamento de projetos de sites web que serão úteis a todos...

### Perguntas a responder antes de iniciar o projeto do site

- Quais são os principais objetivos do seu site? (Informar? Vender? Dar suporte?)
- Quem é o seu público-alvo primário e secundário? (descreva interesses, necessidades, habilidades)
- Qual é a capacidade dos usuários? (browser, velocidade de acesso)
- site atrairá diferentes pessoas? Quais? Quais são as suas áreas de interesse?
- Qual é o principal slogan do site? (a mensagem que melhor descreve seu conteúdo)
- conteúdo do site é novo ou já existe em outro formato?
- Que imagens já existentes estão disponíveis?
- É necessário obter dados dos clientes? O quê é preciso saber? Porquê?
- Que novas tecnologias serão utilizadas? Quais e, especificamente, porquê?
- Que informação do site mudará? Com que frequência e com que abrangência?
- Qual é a posição do produto? (compare com os concorrentes)
- Descreva o produto como se fosse uma pessoa. (sério? estranho? jovem? confiável?)
- Que áreas do site precisam de atualização? Quem se beneficia com esta atualização?
- Pontos fortes e fracos do produto. (compare com os concorrentes)
- Quando o site precisa estar pronto?
- Quem aprovará o trabalho?
- Quem hospedará e dará manutenção ao site?
- site funcionará durante quanto tempo?
- Qual o orçamento para o site?
- Quais são os planos para promover o site? Quem é responsável pela promoção?

### As treze armas secretas de quem constrói sites

**Segmentação do Mercado.** Saiba qual é o seu público, conheça os desejos de seus clientes e faça deles o seu objetivo.

**Visão do Negócio.** Muitos projetistas de sites não sabem quais os objetivos de negócio dos seus clientes. Pense em formas de usar a web de forma eficaz para a empresa.

**Saber Dizer Não.** Se um cliente não se encaixa nos seus critérios, analise se vale a pena aceitar o trabalho. Como grandes projetos são geralmente melhores que pequenos projetos, é melhor deixar de lado alguns projetos pequenos para concentrar-se em um projeto maior.

**Termo de Compromisso.** Este documento declara o compromisso de desenvolver o projeto, dando suas linhas gerais. É fundamental tê-lo mesmo nas primeiras discussões, antes de se chegar a um contrato. Se você está trabalhando sem ter ao menos um termo de compromisso, está indo rápido demais...

**Site do Projeto.** Tenha um site web sobre o projeto de cada site que você está construindo, contendo todo tipo de informação útil (objetivos, cronograma, tarefas realizadas, etc.). Isto é uma forma eficaz de trabalhar e de se comunicar com o seu cliente. A forma organizada de trabalhar pode até ajudar a conquistar novos clientes.

**Estratégia.** A fase de planejamento estratégico é a mais importante do projeto. Conheça detalhadamente o negócio do seu cliente. Clientes que levam a Internet a sério diferenciam os bons dos maus projetistas pela qualidade de sua estratégia.

**Perfil do Usuário.** Saiba tudo sobre o seu usuário: idade, local onde mora, profissão, tipo de família, revistas que lê, tempo que passa na Internet, hobbies, etc. Imagine que você fosse contratado para construir um site que tem como público-alvo apenas 4 pessoas. Não valeria a pena descobrir, individualmente, o que fazer para que eles voltassem a visitar seu site? Mesmo que o público-alvo incluía milhões de pessoas, você deve agir da mesma forma...

**Envolvimento do Líder.** Em cada projeto, existe uma pessoa na empresa que lhe contratou que funciona como um líder, um ponto focal do cliente. Se esta pessoa está envolvida no projeto, todos são beneficiados. Leve-o a acompanhar todo o processo de desenvolvimento, como se fizesse parte da sua equipe.

**Comunicação.** A maior causa para que os projetos acabem mal está na falta de comunicação ou na má comunicação. A natureza mutante da Internet e a rapidez na liberação de novas versões de browsers podem confundir ou frustrar qualquer um. Às vezes, basta um mal entendido em torno de um simples termo técnico. Clientes e projetistas devem conversar pelo menos duas vezes por semana.

**Tratar Clientes Individualmente.** Cada cliente precisa de atenção e envolvimento pessoais. Se você está recebendo e-mails de mais de cinquenta clientes por dia, está na hora de repensar seu trabalho.

**Prazo para "fechar" o Conteúdo.** Se é um projeto grande, insista em não incluir mais nenhum conteúdo nas últimas duas semanas antes do lançamento do site. Mesmo em projetos pequenos, não deve ser menos de uma semana. A única possível exceção são as seções de "novidades" de algumas páginas.

**Número de Projetos.** Cada grande projeto deve ter um produtor dedicado, mas muitos estão envolvidos em vários outros projetos. Para isso, o produtor deve ser extremamente organizado. Um bom produtor conseguirá dar conta, no máximo, de dois grandes projetos. Mesmo assim, apenas se os prazos dos dois não forem próximos um do outro.

**Gerenciamento de Conteúdo x Gerenciamento de Projetos.** Um projetista utilizará uma ferramenta de gerenciamento de projetos para criar o site. O cliente utilizará uma ferramenta de gerenciamento de conteúdo para fazer a atualização diária do site, criando documentos segundo os modelos definidos durante o projeto.

## As sete falsas economias nos projetos de sites

"O nosso site é uma versão na Internet do nosso marketing atual. Podemos usar os mesmos textos e gráficos de nossos folders, anúncios de jornal ou revista."

"Se mais pessoas trabalharem no site, ele ficará pronto mais rápido!"

"A web é um lugar onde nós podemos fazer várias experiências para depois ver o que deu certo. Vamos contratar aquele amigo nosso que faz sites nas suas horas vagas."

"Documentação é uma perda de tempo e esforço..."

"Vamos fazer nós mesmos o projeto do site. Depois contratamos alguém para pegar os nossos rascunhos e transformá-los em páginas web."

"Vamos pedir propostas de construção do nosso site a alguns projetistas e agências de propaganda. Assim, teremos um monte de boas idéias de graça!"

"Sites web não precisam de manutenção. Vamos até nos divertir fazendo tudo por conta própria..."

## O cliente tem direito a

- Cronogramas bem feitos.
- Comunicação constante.
- Um contrato por escrito.
- Notificação imediata em caso de atrasos, problemas e despesas extras.
- Pagar somente pelo que foi autorizado.
- Controle de horas, quando está pagando por hora.
- Ver o projeto durante o desenvolvimento.
- Tempo de resposta razoável.
- Precauções relativas a segurança e privacidade.
- Produtos que funcionam de acordo com o que foi contratado.
- Direitos autorais sobre o conteúdo do site.

## O projetista de sites tem direito a

- Saber qual o orçamento do projeto, desde o início
- Um perfil completo do projeto.
- Um contrato por escrito.
- A valorização do seu tempo (atrasos por parte do cliente custam dinheiro).
- 25-50% do valor orçado como pagamento inicial.
- Pagamento no prazo combinado.
- Gerenciar seu próprio processo.
- Compensar suas falhas.
- Ser pago por trabalhos de consultoria ou idéias.
- Ser informados quando estão fora da concorrência por um projeto
- Direitos autorais sobre as ferramentas criadas para construir o site.

## Pontos chaves para fechar um projeto

- Tenha um site web informativo que diferencie a sua empresa de projetos de sites web da concorrência.
- Tenha um panfleto ou folder com informação sobre o seu trabalho e o testemunho de clientes.
- Seja a pessoa com quem seu cliente queira fazer negócio, sendo profissional e capacitado.
- Certifique-se de que deseja fazer negócios com seu cliente. É uma via de mão dupla.
- Aprenda sobre o negócio do seu cliente.
- Indique clientes anteriores que estão plenamente satisfeitos com o seu trabalho.
- Não vá abaixo do seu preço mínimo, a não ser que tenha de fazê-lo ou queira fazê-lo.
- Não entregue uma proposta ou informação específica sobre o seu projeto antes de ter certeza que o cliente está realmente interessado em contratá-lo (e que não está contactando mais de três outras empresas).
- Faça boas estimativas.
- Não importa se você gosta ou se o cliente gosta. A única coisa que importa é se o usuário do site vai gostar.
- Se você está pronto para fechar o negócio, não saia sem um termo de compromisso.

## Dez segredos para produzir sites web

- Use uma nomenclatura consistente.
- Padronize os tamanhos dos gráficos criando uma hierarquia: títulos, subtítulos, gráfico grande, gráfico médio, gráfico pequeno, amostras, etc. Use o menor número de classes possível.
- Use uma palette otimizada para as imagens em cada classe. Use sempre a mesma palette para uma determinada classe.
- Insira comentários nas suas páginas HTML, de forma a dar informações para os projetistas e pessoal responsável pelo conteúdo.
- Use exatamente os mesmos comandos em páginas diferentes para facilitar futuras buscas e substituições.
- Em páginas grandes, remova os comentários e elimine saltos de linha na versão final que será publicada no site.
- Use o máximo de recursos que sua ferramenta oferece e atualize a versão desta ferramenta com frequência.
- Faça cópias de segurança do seu site.
- Verifique como o seu site aparece tanto no Microsoft Internet Explorer quanto no Netscape Navigator. Se possível, teste tanto nas versões 3.0 quanto nas versões 4.0 destes dois browsers.
- Teste, teste e depois teste tudo de novo. Tenha sempre uma versão alfa ou beta disponível para que seja possível fazer testes e dar opiniões.

---

## SITES DE 3a. GERAÇÃO

A Web evoluiu muito nos últimos anos. Estamos agora na versão 4.0 dos browsers (Microsoft Internet Explorer e Netscape Navigator) e as páginas Web evoluíram no mesmo ritmo. Entretanto, a classificação dos sites em gerações não tem relação direta com os recursos oferecidos pelos browsers. Os requisitos para ser classificado como um site de terceira geração estão mais relacionados ao design do que propriamente à tecnologia...

### A primeira Geração

Os primeiros sites na Web privilegiavam apenas o conteúdo e não a forma. Foram criados por cientistas que desejavam compartilhar suas idéias com outros cientistas. Eles eram estritamente lineares e tinham um mínimo de funcionalidade.

Na primeira geração, nota-se claramente a limitação imposta por modems lentos e monitores monocromáticos. Os gráficos e textos eram apresentados sempre de cima para baixo e da esquerda para a direita. Era muito comum o uso de saltos de linhas, marcadores e linhas horizontais como recursos para separar parágrafos.

### A segunda Geração

No início de 1995, foram lançadas diversas extensões à linguagem HTML no browser Netscape Navigator (que então dominava sozinho a web). A evolução nos sites surgiu na forma de ícones, imagens de fundo, botões com bordas, tabelas e gráficos mapeados. A estrutura deixa de ser linear para ser apresentada de forma hierárquica, quase sempre através de menus com vários níveis.

A maior diferença da segunda geração para a primeira foi a substituição de palavras por elementos gráficos. As funções passam a ser representadas por ícones, surgem imagens de fundo ao invés dos antigos fundos cinzas, os gráficos coloridos e animados substituem as antigas figuras. Cria-se o conceito de "home-page": uma página cheia de desenhos 3D, janelas e botões, que serve de menu para acessar o restante de um site.

Nesta época, a legibilidade deixou de ser importante. Para ter um bom site, era necessário mostrar uma grande quantidade de truques técnicos.

### Sites de Terceira Geração

O que diferencia a terceira geração das demais não são os recursos tecnológicos. A grande diferença está no design. A idéia é dar aos usuários uma boa sensação. As pessoas mais habilidosas conseguiram criar sites de terceira geração usando qualquer browser gráfico.

Na terceira geração, o conteúdo volta ao seu lugar de destaque. Entretanto, a forma não é mais deixada de lado. Há uma preocupação simultânea com funcionalidade e beleza estética. Há grande preocupação no layout preciso, na harmonia entre as cores, na escolha do tipo de letra adequado, no uso correto dos gráficos e no tempo para carregar cada página. Acima de tudo, há um compromisso de ser agradável (em todos os sentidos) ao usuário.

Os projetistas destes novos sites utilizam metáforas e modelos psicológicos dos consumidores. Assim como os arquitetos que fazem os shopping centers, eles passam

horas e até dias pensando em como tornar suas páginas mais atraentes aos seus usuários.

Criar sites de terceira geração é um trabalho árduo, que exige dedicação e uma grande sentimento daquilo que agrada o seu público-alvo. Em geral envolve o trabalho de uma equipe que precisa trabalhar unida para fazer cada página ser bonita e o site como um todo funcionar como uma boa experiência para o usuário.

## O Restaurante

É interessante a comparação de um site web com um restaurante.

Você descobre um restaurante através de um amigo, de um anúncio ou passando por acaso por ele. Você lê cartazes ou faixas com ofertas do lado de fora, pára na entrada e sente o clima e o cheirinho da comida.

Estando na porta, você decide se vai ou não entrar. Em um restaurante popular, você até pode esperar em uma fila para conseguir uma mesa. Se você ficar, será levado a uma mesa e lá lhe mostrarão um cardápio. Você faz a sua escolha.

Quando a comida chega, você aprecia o prato. Tanto a comida como a sua arrumação no prato são obra do chef. Você prova um ou outro item, mistura alguns para experimentar o sabor.

No final, você escolhe uma sobremesa, pede a conta e paga. Você deixa uma gorjeta e pode até conversar um pouco com o maitre ou o dono. Depois, quando sentir fome novamente, poderá voltar ou não, de acordo com a qualidade da sua primeira experiência.

Pense na Web como uma cidadezinha aconchegante com meio milhão de restaurantes.

## Quarta Geração?

Muitas pessoas acreditam que os sites de hoje são de quarta geração, pelo simples fato de terem sido projetados para visualização com a versão 4 dos browsers. Outros acreditam que a quarta geração são os sites com páginas dinâmicas e acesso a banco de dados.

Nada disto é verdade. O que define a geração de um site não é a tecnologia usada para construí-lo, mas o seu design. Um bom projetista pode criar um site de terceira geração que possa ser visualizado com o Netscape 1.1. Um site que explora recursos do Internet Explorer 4.0 pode ser de segunda geração.

## Páginas de Entrada

Uma característica típica de um site de terceira geração é a página de entrada. Ao invés de mostrar diretamente ao usuário o seu menu de opções, você o convida a ver uma página inicial, uma porta de entrada ou uma "Splash Screen". Em alguns sites, existe uma seqüência de páginas de entrada, formando um túnel ou corredor.

A idéia é simples: fazer com que o usuário sinta o que encontrará dentro do site. O usuário deve sentir-se atraído a continuar, seja por estímulos positivos ou por um certo grau de suspense.

Algumas páginas usam túneis (várias páginas de entrada), mas neste caso o risco é grande. Se não for muito bem elaborado e atraente, um túnel poderá afastar usuários ao invés de atraí-los. Nunca use mais de quatro páginas. Tenha um link direto para a página principal para o caso do usuário não desejar passar pelo túnel.

Um defeito básico a evitar nas páginas de entrada é um tempo grande para carregá-las. Se a sua página de entrada demorar mais de 15 segundos para aparecer (levando em conta a velocidade mais comum dos modems), o seu usuário poderá nunca chegar ao seu menu de opções. Pelo contrário, ele acabará procurando uma página mais interessante no C@dê.

Outro defeito é tentar fazer o usuário se registrar logo na entrada. Frases do tipo "registre-se aqui de graça" não funcionam mais. Se você deseja realmente que as pessoas se registrem, você deve dar algo a elas, antes. Um registro na sua página ou túnel de entrada significa uma barreira que vai apenas espantar a maioria dos seus usuários.

## Página Principal

Ao contrário da geração anterior, os sites de terceira geração podem ter uma ou várias páginas básicas (home-pages), como uma forma de organizar ou apresentar o seu conteúdo. Alguns sites simplesmente não possuem uma página principal.

Na terceira geração, as páginas principais devem ter conteúdo, além de servirem como ligação com as demais páginas. Não tenha medo de orientar o seu visitante. Inclua vários links para outras páginas do site em cada página. Tenha sempre algo interessante em cada página.

Se o seu site vende produtos, deve haver um link para o catálogo em quase todas as páginas do site. Se você tem um formulário online para venda do seu software, tenha um link para ele em todas as páginas que falarem deste software. Se você deseja que o usuário preencha uma pesquisa, faça com que o link apareça em cada página. A maioria das pessoas não vai clicar nestes links na primeira vez, mas eles acabarão clicando quando estiverem prontos.

## Iscas

Utilize iscas para atrair os visitantes ao seu site. Fofocas, notícias, promoções de vendas, software grátis, arquivos com sons, fotos da loura do tchan e receitas culinárias são coisas que atraem usuários para sites de terceira geração. Se você quer atrair donos de cachorros, crie uma seção "anatomia de uma pulga" ou "catálogo de raças de cães". Se você quer atrair amantes de cinema, tenha uma lista dos filmes mais alugados na semana.

Na web, estas iscas costumam ser chamadas de "coisas grátis" (free stuff). Se você der coisas grátis, mais usuários passarão pelo seu site. Use a imaginação! Pense em alguma coisa que o seu público alvo goste de ver, ouvir ou falar. Quando os usuários começarem a dar o endereço do seu site para os amigos, a coisa estará dando certo. Porém, cuidado! Quanto mais coisas grátis você der, mas os usuários vão querer. Prepara-se, então, para continuar dando sempre mais coisas grátis.

## Páginas de Saída

Ao contrário do que possa parecer à primeira vista, um link para a saída do seu site não fará os usuários irem embora. Mostrar uma página (ou túnel) de saída faz com que os usuários sintam que estão saindo e reflitam: "será que eu já visitei tudo o que queria neste site?".

Anunciar a saída também cria uma certa expectativa. Vale a pena perder um pouco de tempo para criar uma saída realmente interessante. Não é bom, entretanto, fazer muito alarde. Deve haver um link sutil, sem destaque a mais ou a menos. Sem encorajar o usuário a sair, mas ao mesmo tempo mostrando que existe esta opção.

A página de saída é também um bom lugar para perguntar algo ao seu usuário. Você pode pedir para que ele preencha um formulário, ligue para o seu número 0800, participe no sorteio de um prêmio, assine uma lista de discussão ou coisas do gênero. Neste ponto, o usuário deve estar satisfeito com a experiência e pode estar disposto a lhe dar algo em retorno.

O "grande final" pode ainda incluir um último comentário sobre o tema, uma lista de sites relacionados na Internet ou similares.

## Mudanças

Se você tem um site, com certeza deseja entrar para o "bookmark" ou a "lista de favoritos" do seu usuário. Se a sua página principal for muito boa, isto pode acontecer. Se as iscas o levaram até lá, ele voltará para checar as novidades. Seu site, então, pode ser beneficiado por mudanças.

Se você planeja mudar seu site uma vez por mês, é melhor nem se dar a esse trabalho. É como se ele fosse estático. Um site que muda semanalmente e tem informações interessantes pode atrair alguns usuários. Páginas que possuem conteúdo novo e atraente diariamente são uma certeza de um grande número de acessos.

O ideal é colocar as novidades em destaque logo na página principal e não criar um link para uma página "novidades". Se as novidades são realmente interessantes, elas merecem um lugar de destaque no seu site.

## Metáforas

Um dos elementos típicos de um site de terceira geração são as metáforas. Criando um site que lembra algo do mundo real, você torna a navegação mais fácil e ajuda a dar uma coerência ao seu site.

Metáforas devem ser conhecidas, consistentes e apropriadas para a velocidade de acesso do seu usuário. Elas fazem o usuário se sentir à vontade, dão a ele um sentimento de que já sabe navegar pelo site. Uma boa metáfora pode também levar o usuário a explorá-lo por inteiro, para descobrir até onde ela pode chegar...

As metáforas mais comuns são galerias, museus, revistas em quadrinhos, lojas, canais de televisão, shopping centers, livros, jornais, estantes, parques de diversão, pessoas, computadores, animais, fazendas, prédios, cidades e todo tipo de construção, ser ou objeto do mundo real.

Metáforas são um meio de exploração. Devem ser simples, consistentes e fáceis de usar. Boas metáforas são óbvias e são construídas de forma a serem intuitivas. Uma metáfora ruim força você a aprender novos conceitos e comandos. Se a metáfora for boa, você não consegue se perder no site.

Algumas metáforas exageram no uso de gráficos tridimensionais. Elas possuem um grande realismo e permitem que o usuário abra portas, desça escadas e passeie por corredores. Embora isto fique ótimo em CD-ROMs e redes locais, este tipo de metáfora acaba sendo uma frustração para o usuário da Internet que acessa via modem. Suas metáforas devem ser leves e eficientes.

As metáforas também devem ser familiares para o seu público-alvo. Um site que se baseia na estrutura de um motor de carro é ótimo para amantes de mecânica, mas pode ser péssimo para outras pessoas.

Se decidir usar uma metáfora, você deve usá-la no site inteiro e não somente numa parte dele. E, uma vez escolhida uma metáfora, mantenha sua linha e seja consistente. Pode parecer fácil, mas você será tentado a relaxar em alguns setores do site ou a estender sua metáfora para outros temas. Resista à tentação e mantenha as coisas simples e coerentes.

Projetistas gráficos são pouco explorados na construção de sites web. Eles possuem grande habilidade para criar metáforas em cartões de visita e comerciais de TV. É melhor interagir com este profissional do que tentar tornar-se um.

## Temas

Você não precisa de uma metáfora para ter um site de terceira geração. Um tema consistente é suficiente. Um tema pode ser visual ou conceitual. Os exemplos mais comuns são: primitivo, fotográfico, infantil, tipográfico, futurista, náutico, entre muitos outros. Assim como uma metáfora, um tema pode ajudar ou atrapalhar.

Quase qualquer coisa pode servir como um tema. Pense nas vitrines das lojas, que normalmente possuem um tema. A maioria usa um conjunto consistente de cores, texturas, iluminação e gráficos. Decoradores sabem muito sobre temas. Eles criam espaços funcionais e interessantes que não são repetitivos. Um decorador sabe fazer ambientes agradarem os sentidos ao mesmo tempo que servem a um propósito.

Sites temáticos são mais difíceis de criar do que parece. Existe uma grande tentação de usar todos os tipos de recursos: som, animação, fontes e gráficos, criando uma grande confusão. O uso de fotografias de qualidade, por exemplo, pode fazer uma grande diferença em um site.

Usar o número reduzido de cores disponíveis no browser já é difícil. Usar somente um subconjunto destas cores para criar um tema é um desafio e tanto. Um bom site temático requer um grande esforço para unir um projeto atraente à consistência de estilo.

## Sites de Informação

Muitos sites não são voltados para consumidores. Sites que possuem um grande volume de informação devem satisfazer a usuários impacientes que querem ir direto ao assunto. Estes sites não podem se dar ao luxo de colocar muitos adornos em torno da informação. Entretanto, eles podem ser agradáveis sem utilizar muitos gráficos.

A maioria destes sites contém longas páginas de texto e listas com marcadores. Seu menu principal é conhecido: Novidades | Nossa Empresa | Índice | Perguntas Comuns | Ajuda. Muitos possuem ferramentas de busca que permitem pesquisar as páginas e ajudar aqueles que não sabem exatamente o que procuram.

A apresentação de informação vinda de um banco de dados é uma tarefa complicada. Sites como o Alta Vista precisam harmonizar em suas páginas os resultados da busca, anúncios, controles de navegação, diferentes níveis de usuários e ofertas tentadoras.

Os projetistas destas páginas devem trabalhar com modelos que são preenchidos no momento de cada pesquisa. O resultado deve ser atraente e funcional, sem confundir o usuário. Quadros (frames) podem ajudar, mas tenha cuidado com a simplicidade.

Os servidores estão fornecendo cada vez mais recursos de personalização. Para sites de informação, isto significa a possibilidade de mostrar ao usuário somente o que lhe interessa. A criação de menus personalizados é um recurso apreciado por todos os usuários. Ofereça a possibilidade de enviar e-mails informando quando uma informação de interesse for adicionada ao site.

## Formulários

Projeto de formulários é outra especialidade. Alguns formulários são bem melhores que outros, como acontece com muitas coisas na web. Alguns deixam o usuário confuso, desorientado e sem saber como pedir o produto. A melhor forma de criar um formulário é procurar bons exemplos na própria web e imitá-los. Não cometa o erro de pensar que formulários são simples de fazer.

Prefira os formulários sem bordas ao redor dos campos. Em geral é uma boa idéia alinhar os títulos à direita e colocar os campos a preencher alinhados pela esquerda. Também não é recomendável criar uma longa página com dezenas de campos. Prefira criar várias páginas com um botão "Próxima Página" fazendo a ligação. Numerar os passos e separar as informações usando cores são boas idéias.

Muitas vezes o seu usuário está impaciente ou simplesmente não deseja fornecer muita informação ao seu site. Peça o mínimo possível de informação e mantenha o formulário simples.

## Conclusão

Se as pessoas começarem a falar do seu site, se elas voltam com frequência, se a sua metáfora for mencionada por alguém ou se a sua página de entrada estiver realmente boa, então você conseguiu criar um site de terceira geração.

As pessoas começarão a ter uma certa identificação com o site. Elas se sentirão realmente atraídas a fazer do site um lugar para visitar regularmente. Você terá uma verdadeira comunidade em torno do seu site.

Se você conseguir tudo isso, terá realmente conseguido criar um excelente site.

## GRÁFICOS PARA WEB

Gráficos são elementos importantes na construção de sites web. Além das preocupações comuns relativas à beleza e criatividade, este tipo de gráfico deve ser criado pensando nas limitações relativas a cores e tamanho do arquivo gerado...

### Vetores x Mapas de bits

Existem basicamente dois tipos de imagens feitas por computador: vetoriais e bitmaps. Vetores são imagens definidas a partir de regras como "desenhe uma linha de 10,10 até 120,30", "desenhe um círculo com centro em 50,60 e raio 10", "pinte a área em 20,20". Os bitmaps são diferentes: você tem o desenho feito ponto a ponto, como se ele fosse pintado sobre papel quadriculado, onde cada quadradinho pode ser pintado de apenas uma cor.

Uma imagem vetorial simples é um arquivo pequeno, que pode ser facilmente desenhada em vários tamanhos sem perder qualidade. As extensões de gráficos vetoriais mais conhecidas são WMF (Windows Metafile), CDR (Corel Draw), DWG, DXF (ambos do AutoCAD) e AI (Adobe Illustrator).

Uma imagem em mapa de bits é mais detalhista, mas geralmente ocupa mais espaço que uma imagem vetorial simples. Este formato é necessário quando se utiliza imagens digitalizadas, mas apresenta problemas para ser mostrada em diversos tamanhos. Este tipo de imagem perde qualidade nos tamanhos muito pequenos e fica grosseira em tamanhos muito grandes. As extensões de gráficos mais comuns são BMP (Windows Paint), PCX (Paintbrush), PSD (Adobe Photoshop), CPT (Corel Photopaint), TIF (usado em editoração), GIF e JPG (usados na Internet).

Na Web, a grande maioria dos gráficos são bitmaps com extensão GIF e JPG. A razão é que estes formatos são bitmaps comprimidos. Os arquivos GIF e JPG são bem menores, por exemplo, que a mesma imagem em formato BMP.

### Tamanho do Arquivo

Existem uma série de fatores que influenciam o tamanho de um arquivo gráfico para a Internet:

**Número de pontos.** Obviamente, a largura e a altura de um gráfico influenciam diretamente o tamanho do arquivo.

**Número de cores.** É necessário guardar informação sobre cores para cada ponto de um bitmap. Em um gráfico que usa cores realísticas, cada ponto ocupa 3 bytes e é possível representar mais de 16 milhões de cores. O mais comum, na Internet, é usar arquivos onde cada cor é guardada em um byte, permitindo usar até 256 cores.

**Compressão.** Os formatos GIF e JPG guardam os bits comprimidos. Esta compressão consiste em achar repetições dentro do arquivo e eliminá-las.

A maioria das pessoas imagina que apenas a dimensão da figura (sua altura e largura em pontos) modificam o tamanho. Como um desenvolvedor de gráficos para a Internet, você deve preocupar-se bastante com a compressão e principalmente com o número de cores de cada figura. Uma mudança nestes itens pode facilmente reduzir seu arquivo à metade do seu tamanho original. Em alguns casos, o tamanho chega a ser reduzido a um décimo.

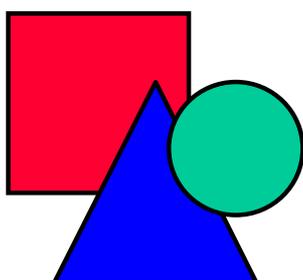
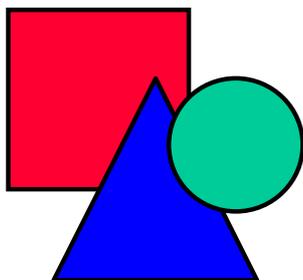
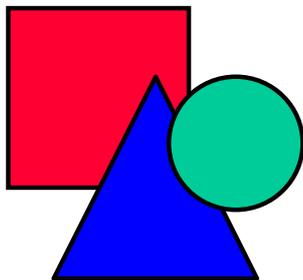
## Cores

Como já dissemos, o número de cores influencia fortemente o tamanho do arquivo. Saber utilizá-las corretamente pode ser muito trabalhoso, mas vale a pena. Não estou sugerindo que você faça apenas figuras em preto e branco, de agora em diante. O uso de várias cores é fundamental para a aparência do site. A mágica está em reduzir o tamanho do arquivo sem perder qualidade e variedade de cores. Acredito, isto é possível!

Inicialmente você precisa entender em detalhes o quanto as cores influenciam o seu arquivo e como elas são realmente armazenadas e tratadas pelo browser. Observe a tabela abaixo, que mostra o tamanho de um arquivo de 100 x 300 pontos com diversos esquemas de cores:

Tipo de arquivo	Número de Cores	Bits por ponto	Tamanho arquivo*
Preto & Branco	2	1	1.250
Colorido, 16 cores	16	4	5.000
Colorido, 256 cores	256	8	10.000
Cor real, 16 bits	32.768	16	20.000
Cor real, 24 bits	16.777.216	24	30.000

Bitmap de 100 x 100 pontos, sem compressão, sem cabeçalhos, sem palette, em bytes



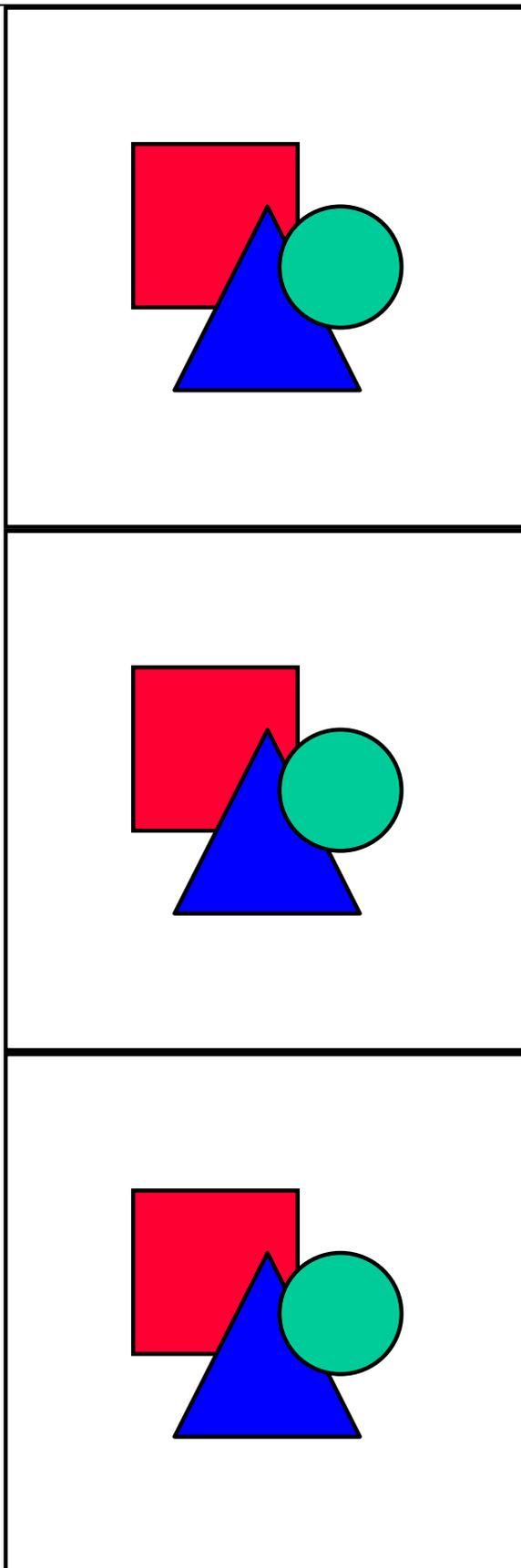
## Palettes

Quando se utiliza cor real, cada ponto tem sua cor definida através dos componentes em vermelho (red), verde (green) e azul (blue). Estas são as cores básicas para formação de todos os milhões de cores no nosso monitor.

Se você já definiu cores de fundo para páginas, links ou tabelas em HTML, deve conhecer o esquema RGB. Cada cor é definida como um número hexadecimal de 6 dígitos. São 2 dígitos para vermelho, dois para verde e dois para o azul. O valor de cada cor vai de 00 até FF (255 em decimal). Cada componente ocupa um byte e permite, portanto, 256 possibilidades. Multiplicando 256 por 256 por 256, temos as 16.777.216 cores disponíveis.

Quando se utiliza um número menor de cores, não vale a pena gastar três bytes para definir a cor de cada ponto. Nem tampouco deve-se utilizar alguns poucos bits para definir a cor, o que deixaria o número de opções muito restrito. A solução é usar uma tabela de cores, adicionada ao início do arquivo, descrevendo as cores a serem utilizadas no arquivo. Esta tabela é conhecida como "palette".

Observe alguns arquivos com palettes de 8 cores (3 bits) e a respectiva palette logo abaixo. Note que, do total dos mais de 16 milhões de cores disponíveis, apenas algumas foram escolhidas para fazer parte de cada palette.



O tipo de arquivo mais comum é o que o utiliza uma palette de 256 cores (8 bits). Isto permite uma grande combinação de cores e uma razoável economia em bytes. Só é

---

recomendável usar mais que 256 cores quando o efeito final precisa ter uma aparência "fotográfica".

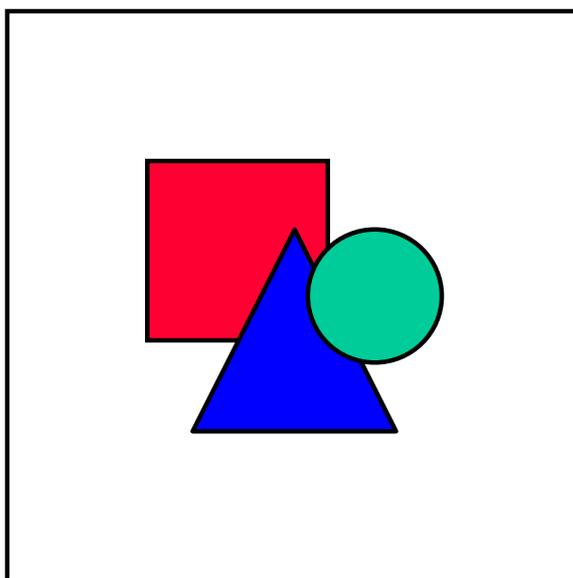
## A Palette dos Browsers

Um dos problemas básicos dos gráficos na web é saber como as cores serão visualizadas pelos usuários. Todo bom projetista de sites possui uma placa de vídeo capaz de mostrar milhões de cores, mas será que o usuário também tem uma placa assim? A resposta, na maioria das vezes, é não.

A maioria dos usuários utiliza sua placa de vídeo no modo de 256 cores. Alguns usuários possuem placas simples, outros não sabem como instalar o driver mais sofisticado e outros preferem 256 cores para melhorar a performance do Windows, especialmente em jogos.

Mesmo usando 256 cores, seu usuário tem algumas destas cores reservadas para o uso do sistema e do próprio browser. Assim, sobram em torno de 216 cores disponíveis para as suas figuras. Se você utiliza gráficos com palettes com muitas cores, é provável que o usuário não consiga ver o gráfico como você o projetou.

O Netscape e o Explorer possuem, inclusive, uma palette padronizada de 216 cores para usar nas páginas e nos seus gráficos. Esta palette inclui todas as cores básicas (verde, vermelho, azul, amarelo, ciano, magenta, preto e branco) e pelo menos 4 graduações entre cada uma destas cores. Veja no gráfico abaixo a palette padrão dos browsers.



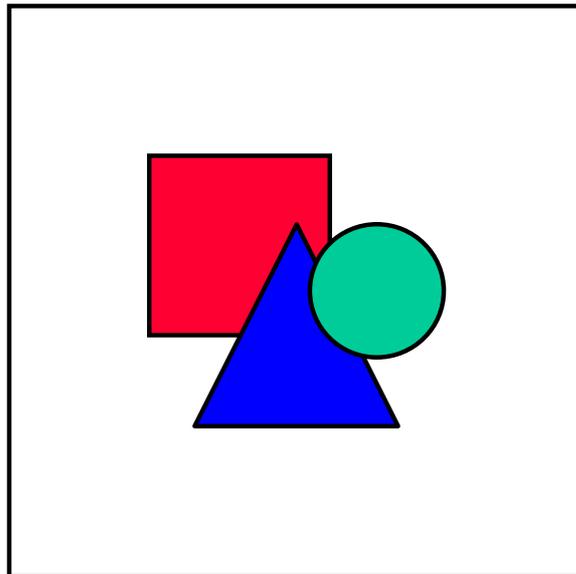
Se você utilizar somente as cores na palette acima, todos os usuários com 256 cores no seu monitor (ou mais) verão sua imagens exatamente como você projetou. Se você utilizar uma palette com mais cores ou cores diferentes das mostradas acima (por exemplo, uma palette com 200 tons de azul), os usuários verão os gráficos de forma diferente, com as cores adaptadas para a palette padrão.

Se você faz gráficos especificamente para a web, é melhor até configurar seu programa de desenho com a palette dos browsers, para ter uma noção precisa de como os internautas comuns verão seus gráficos.

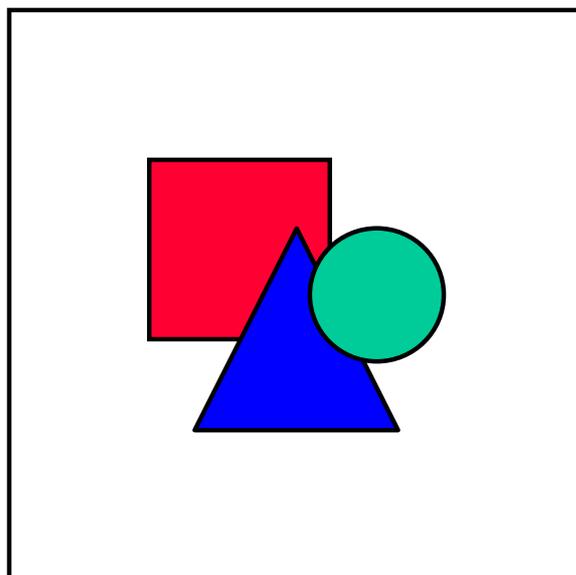
## Anti-aliasing

Um terrível efeito colateral no uso de arquivos bitmap é o efeito de "escadinha" que se tem quando o gráfico inclui linhas inclinadas ou curvas. Os americanos chamam este efeito de "aliasing".

No gráfico abaixo, fica claro que as linhas horizontais não apresentam este efeito. O maior problema está nas linhas inclinadas, nas curvas e nos textos. O problema no texto é ainda maior quando as fontes usadas são mais complexas.



Uma forma de evitar este problema é usar somente linhas horizontais e verticais, evitando linhas inclinadas e curvas. Obviamente, isto não é nada prático. Outra solução, mais realista, faz parte dos recursos de alguns programas gráficos e chama-se "anti-aliasing". Este processo consiste em usar cores intermediárias para suavizar as linhas. No Photoshop, por exemplo, quase todas ferramentas podem funcionar com "anti-aliasing". Observe, na figura abaixo, que foram usados diversos tons de cinza para suavizar o efeito "escadinha".



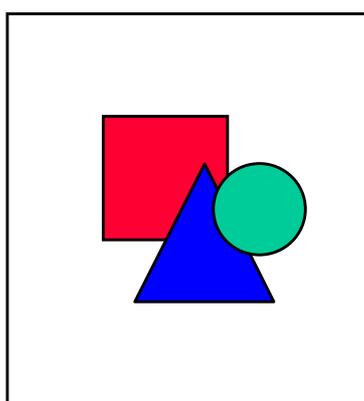
Anti-aliasing exige um número maior de cores na palette. Entretanto, todos devem concordar que vale a pena. Quanto for utilizar anti-aliasing, evite sobrepor objetos de cores diferentes. Quando isso acontece, o número de cores necessárias para o anti-aliasing é maior.

## GIF

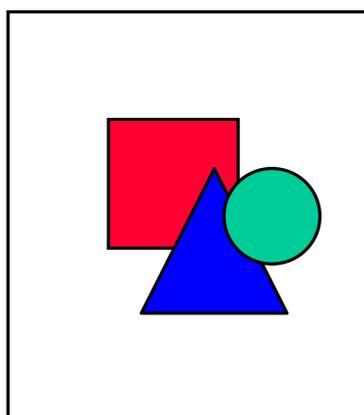
O formato de arquivos GIF (Graphics Interchange Format) foi criado pela Compuserv, um dos maiores serviços on-line dos EEUU (algo parecido com as antigas BBS). Ele é um formato de bitmaps, trabalha com palettes e inclui uma conversão conhecida como LZW, que é muito eficiente para determinados tipos de arquivos.

A compressão do arquivo GIF consiste em codificar linhas de pontos que possuem várias cores repetidas em seqüência. Se uma linha, por exemplo, possui 10 bits na cor 30, ela pode ser compactada. Normalmente, a linha seria gravada como "30, 30, 30, 30, 30, 30, 30,30, 30, 30". Depois de compactada, ela seria transformada em algo como "10 x 30". Obviamente, os bits gravados no arquivo não são exatamente como no exemplo acima, mas a idéia é a mesma.

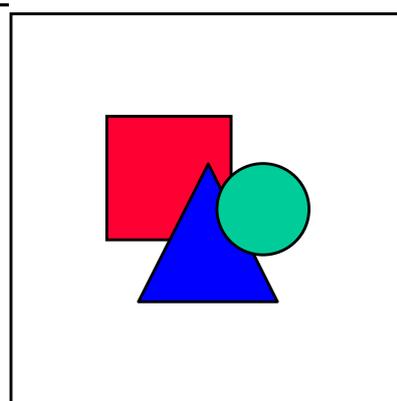
Veja alguns exemplos reais de como se comporta a compressão dos arquivos GIF observando as figuras abaixo, que possuem 100 x 100 pontos cada uma, com palette de 256 cores.



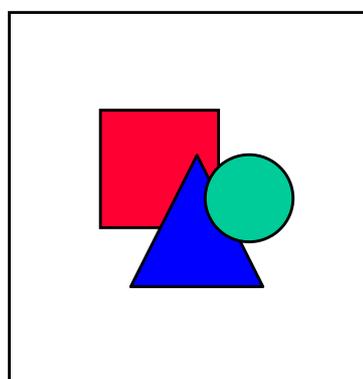
**BMP, 11.080 bytes**



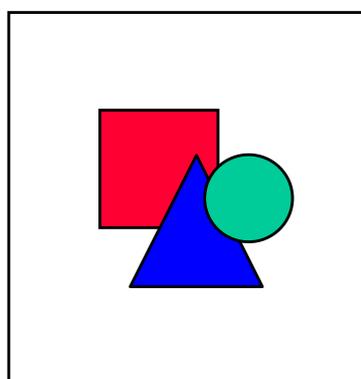
**GIF, 956 bytes**



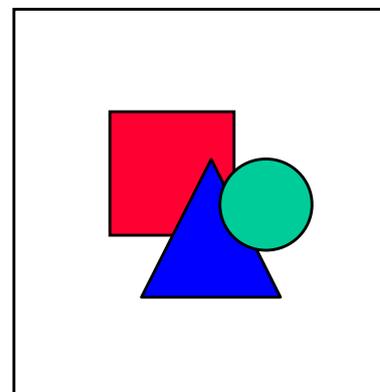
**GIF, 1.023 bytes**



**GIF, 1.121 bytes**



**GIF, 1.388 bytes**

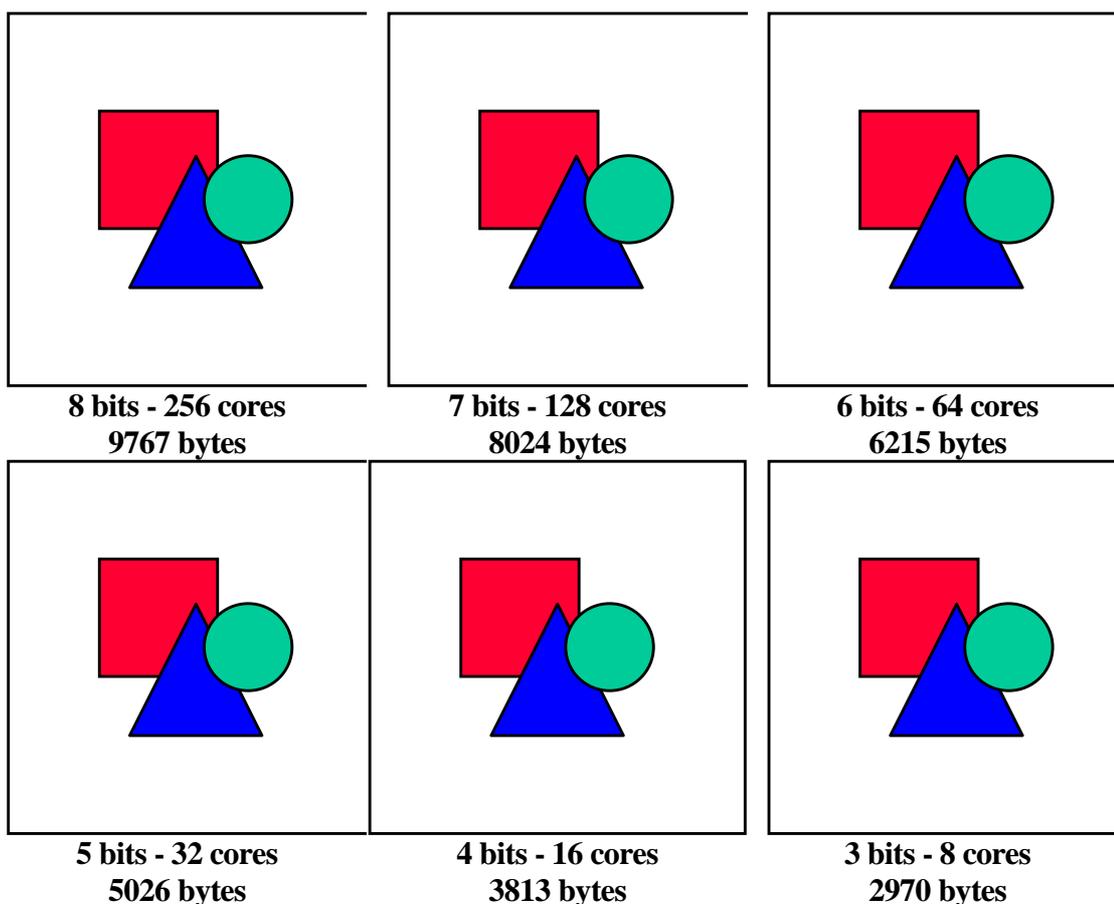


**GIF, 8.502 bytes**

O maior arquivo é o BMP, que não usa compressão. Entre os GIFs, você percebe que os mais compactáveis são os que possuem grandes linhas horizontais.

## Redução de Palette de GIFs

Outra forma de comprimir arquivos GIF é utilizar uma palette menor. Isto reduz tanto o cabeçalho do arquivo (a palette em si) quanto o número de bits necessário para armazenar os pontos propriamente ditos. A economia com a redução de palettes pode ser considerável. Para isso, é preciso um software que consiga reduzir as cores criando uma nova palette e ajustando a figura. Observe a economia de espaço e as conseqüências na figura, através dos exemplos abaixo:



Com a redução da palette de 8 bits (256 cores) para 3 bits (8 cores), diminuimos o arquivo para apenas 30% do seu tamanho. É bem verdade que a imagem de 8 cores ficou bem pior do que o original. A imagem de 32 cores, entretanto, ficou ainda muito boa e nos dá uma economia de bytes da ordem de 50%. Será que não vale a pena?

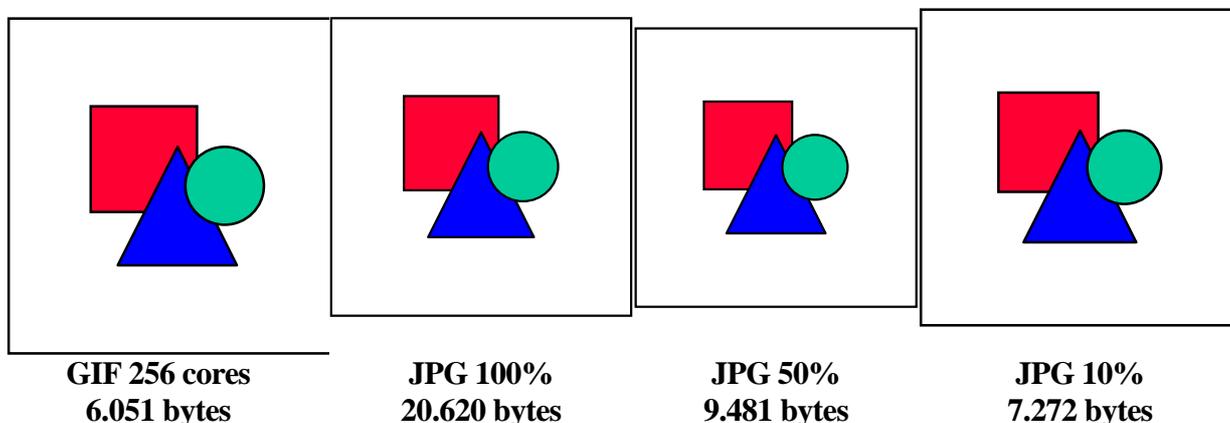
A principal regra na redução de palette é ir diminuindo o número de cores até que se note uma significativa perda de qualidade. Neste ponto, você desfaz a última redução e salva o arquivo.

## JPEG

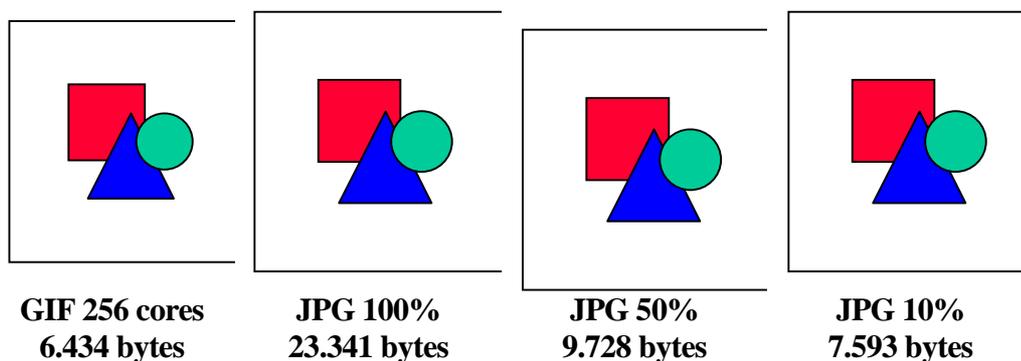
O formato JPEG (Joint Photographic Experts Group) ou JPG, é adequado para a compressão de arquivos com fotos ou desenhos com muitos detalhes. Ele pode ser configurado em diferentes níveis de compressão que vão de 10 a 100. No nível 100, a

imagem obtida do arquivo JPG é idêntica ao original. Nos outros níveis, a imagem é ligeiramente prejudicada, embora ainda seja muito semelhante.

Veja os exemplos abaixo e compare a eficiência do JPG. O arquivo original, um BMP de 100 x 100 pontos com 16 milhões de cores, ocupa 30.056 bytes. Observe o resultado da conversão para GIF e vários tipos de JPG.



Os ganhos de qualidade são mais facilmente percebidos em fotos do que em desenhos. Observe como uma foto fica quando gravada nos formatos GIF e JPG. O arquivo original era um BMP de 100 x 100 pontos com 16 milhões de cores, ocupando 30.056 bytes.



Note que o JPG 100% tem excelente qualidade. Mesmo o JPG 10% tem uma qualidade superior ao GIF, que tem um certo ar "artificial". O JPG, ao perder qualidade, deixa a figura um pouco "borrada". Isto é aceitável para fotos, mas nem tanto em gráficos detalhados. O GIF deixa a figura com um ar "granulado", que fica melhor em gráficos do que em fotos.

## Ferramentas para Compressão

Existem diversas ferramentas para reduzir o tamanho de seu gráfico para a web. As mais interessantes são:

**Adobe Photoshop** - Este é, provavelmente, o software mais utilizado para editar gráficos bitmap profissionais. Ele possui opção para ler a grande maioria dos formatos de bitmap e gravá-los como GIF ou JPG. Para usar a compressão de JPG, basta usar o menu que aparece quando você pode para salvar um JPG. Para comprimir um GIF, converta-o inicialmente para RGB usando a opção de menu "Imagem > Mode >

RGB". Em seguida, use a opção de menu "Imagem > Mode > Indexed Colors...", selecione a palette "Adaptative" e o número de bits (de 3 a 8). Depois salve como GIF.

**Corel PhotoPaint** - Este software, parte do pacote Corel Draw!, também é ótimo para editar bitmaps. Ele pode gravar arquivos GIF ou JPG. O menu de compressão de arquivos JPG aparece logo após o diálogo de "Salvar...", como no Photoshop. Para reduzir a palette de um GIF, use o comando "Imagem > Converter para... > Cor da paleta", selecione a opção "otimizada" e indique o número de cores (8, 16, 32, 64, 128, 256), salvando como GIF no final.

**GIF Lube** - Este site na Internet presta um serviço muito interessante. Acessando-o, você pode fornecer a URL de uma imagem na web (ou no seu disco local) e convertê-la nas diversas palettes reduzidas de GIF ou nos diversos níveis de JPG. Tudo é muito simples e prático. No final, você escolhe a imagem mais adequada e usa o próprio browser para salvar a imagem reduzida. Se você anda sem paciência para ficar selecionando opções no Photoshop ou no Photopaint, esta é a sua melhor opção...

## Conclusão

Criar gráficos no computador é uma tarefa complicada. Na Internet, existem algumas dificuldades extras, como as restrições de tamanho do arquivo e palette. Entretanto, grande parte da beleza de um site vem de seus gráficos. Por isso, vale a pena estudar bem o tema e aplicar ao máximo os recursos disponíveis...

## **IDC - INTERNET DATABASE CONNECTOR**

**O que é**

**Itens a verificar no servidor**

**Criando uma consulta**

**Formulários com IDC**

## O Que é

O IDC (Internet Database Connector) é um padrão criado pela Microsoft para facilitar a criação de aplicações Web que acessam um banco de dados. Ele funciona de forma simples e você pode criar uma aplicação facilmente. Como Banco de Dados e Internet são aplicações da moda, o assunto é interessante.

O IDC não é coisa nova. Ele funciona desde a primeira versão do servidor Internet da Microsoft. É importante lembrar que existe a forma mais moderna de fazer aplicações de banco de dados, usando as páginas ativas ASP (Active Server Pages). Também não se deve esquecer que esta é uma solução Microsoft, que roda apenas em Windows NT.

Para utilizar este recurso, você precisa ter:

Servidor Windows NT

Internet Information Server (a partir da versão 1.0)

Drivers de ODBC instalados no servidor

Um fonte de dados ODBC (SQL Server ou Access MDB, por exemplo)

No lado do cliente, você pode utilizar qualquer browser (inclusive da Netscape).

## Itens a verificar no seu servidor

Para utilizar os recursos do IDC em suas homepages, você precisa atender a todos os requisitos acima. Uma situação ideal é utilizar o Microsoft FrontPage ou o Microsoft Visual InterDev para editar as páginas, mas isso não é necessário. Você pode chegar lá apenas usando o Notepad.

Antes de mais nada:

Certifique-se que o seu servidor roda Windows NT Server

Certifique-se que você possui o Microsoft Internet Information Server (IIS) instalado no seu servidor. O Windows NT 4.0 vem com a versão 2.0 do IIS, que é suficiente. Se você usa uma versão mais antiga do NT, procure na Internet (<http://www.microsoft.com/iis>) o IIS. Você não precisa do IIS versão 3 ou 4 para usar o IDC

Certifique-se que o driver de ODBC que você vai utilizar está instalado e funcionando. Se você optou por usar o Access, instale uma versão 7.0 (95) ou superior no Servidor. Isto garantirá a presença do driver ODBC e facilitará a criação dos seus bancos de dados. Se você preferir utilizar o SQL Server, instale-o no servidor (ou em outro na rede) e garanta que o driver ODBC está instalado.

Crie um banco de dados e as tabelas que utilizará na sua aplicação. Crie uma "Fonte de Dados" ODBC para este banco de dados. Isto é feito usando a opção "ODBC" do "Painel de Controle" do Windows NT. Se for usar o SQL Server, crie um usuário com direito para acessar este banco de dados.

Crie um diretório para guardar as consultas IDC no seu servidor. Inclua este diretório na lista do IIS, dando direito de "Read/Leitura" e "Execute/Execução". Os arquivos IDC só podem ser rodados a partir de um diretório com o direito de "Execução". Crie um arquivo "default.htm" ou desabilite a opção que permite aos usuários ver o diretório, para aumentar sua segurança neste diretório.

Garantidos os itens acima, você pode criar sua primeira consulta.

## Criando uma consulta

Para ter uma consulta, basta criar dois arquivos no seu diretório especial no servidor. Um arquivo terá a extensão IDC e incluirá o comando SQL da consulta. O outro arquivo, que contém o formato dos dados, terá a extensão HTX (template).

A melhor opção é editar estes arquivos com o FrontPage. Assim, você terá uma interface tipo "Wizard" para criar seus arquivos. Entretanto, os arquivos são simples e podem ser facilmente criados com editores mais simples. Até o Notepad serve. O arquivo IDC contém apenas algumas poucas linhas com comandos. O arquivo HTX é um arquivo HTML com alguns recursos a mais.

Abaixo está a listagem do arquivo CONSULTA.IDC, que funciona como exemplo:

**Datasource: Web FAQ (nome do DSN criado no servidor internet/intranet)**

**Template: consulta.htx (esse é o nome do arquivo destino)**

**SQLStatement: SELECT Codigo, Nome, Telefone FROM Cadastro (pode ser colocado em varias linhas com o + na frente)**

**Username: usuario**

**Password: senha**

Neste exemplo, "LocalServer" é o nome da fonte de dados (este nome é comum para o servidor SQL no NT ou um apelido para uma conexão da rede). O arquivo de formato é especificado como "consulta.htx". Em seguida temos o comando SQL (Liste os campos Codigo, Nome e Telefone da tabela Cadastro). Por último temos o nome e a senha para o acesso ao banco de dados.

Como está especificado no arquivo IDC, precisamos de um arquivo HTX para formatar a saída gerada pelo comando. Este arquivo, com a extensão HTX, inclui uma parte inicial (cabeçalho), a parte de formato das linhas da tabela (detalhe) e uma parte final (rodapé). A divisão destas áreas é feita pelas marcas especiais <%begindetail%> e <%enddetail%>. A parte entre estas duas marcas será repetida, uma vez para cada linha da tabela.

Outras marcas especiais no arquivo HTX permitem incluir os dados dos campos. Para incluir os campos mencionados no exemplo acima, por exemplo, basta incluir as marcas <%Codigo%>, <%Nome%> e <%Telefone%>. Veja no arquivo abaixo um exemplo, o arquivo CONSULTA.HTX:

```
<h1>Relatório do Cadastro</h3>
```

```
<table>
```

```
<tr>
```

```
<th>Codigo</th>
```

```
<th>Nome</th>
```

```
<th>Empresa</th>
```

```
</tr>
```

```
<%begindetail%>
```

```
<tr>
```

```
<td><%Codigo%></td>
```

```
<td><%Nome%></td>
```

```
<td><%Empresa%></td>
```

```
</tr>
```

```
<%enddetail%>
```

```
</table>
```

```
<p>Final da tabela</p>
```

## Formulários com IDC

Uma forma interessante de usar o IDC é utilizar um formulário para informar dados. Suponha que você deseja consultar apenas os dados de um determinado estado. Você pode usar uma página HTML com um formulário e passar os dados do formulário para uma consulta IDC. Veja o exemplo.

Arquivo ESTADO.HTM

```
<html><body><h1>Consulta de Estado</h3>
<form action="estado.idc" method="POST">
<p>Digite o estado:
<input type="text" size="2" name="Estado"></p>
<input type="submit" name="Botao1" value="Consultar">
</form></body></html>
```

Arquivo ESTADO.IDC

**Datasource: Web XXX**

**Template: consulta.htx**

**SQLStatement: SELECT Codigo, Nome, Telefone FROM Cadastro WHERE**

**Estado='%Estado%'**

**Username: Usuario**

**Password: senha**

Podemos usar para o ESTADO.IDC o mesmo CONSULTA.HTX que usamos antes. A grande diferença está no uso do campo %Estado%, que veio do formulário HTML. Ele é inserido no meio da sentença SQL, normalmente.

## ACTIVE SERVER PAGES

**O que é**

**Vantagens do ASP**

**Itens a verificar no servidor**

**Perguntas comuns sobre ASP**

**Tutorial de ASP**

## O que é

As ASP (Active Server Pages - Páginas de Servidor Ativas) são um ambiente para programação por scripts no servidor, que você pode usar para criar páginas dinâmicas, interativas e de alta performance. Como as páginas ASP, os scripts rodam no servidor e não no cliente. É o próprio servidor que transforma os scripts em HTML padrão, fazendo com que qualquer browser do mercado seja capaz de acessar um site que usa ASP.

Entre os recursos que podem ser implementados via ASP, podemos citar:

Programação em VBScript ou JScript

Acesso a banco de dados

Sessões (persistência de informações no servidor)

ASP surgiu juntamente com o lançamento do Internet Information Server 3.0. Esta é uma solução Microsoft, que exige que o seu servidor precisa rodar um sistema operacional da Microsoft (Windows 95 ou NT). Os seguintes servidores suportam o uso de páginas ASP:

Microsoft Internet Information Server versão 3.0 no Windows NT Server

Microsoft Peer Web Services versão 3.0 no Windows NT Workstation

Microsoft Personal Web Server no Windows 95 ou Windows 98

A grande vantagem, porém, é que existe esta exigência apenas do lado do servidor. No lado do cliente, você pode utilizar qualquer browser, mesmo os que não suportam VBScript (como os da Netscape).

## Vantagens do ASP

**Independência do Browser:** ASP pode rodar páginas complexas no servidor e enviar somente os resultados para o cliente.

**Páginas x Bancos de Dados:** Permite visualizar, atualizar e adicionar informações nos servidores SQL

**Segurança do Código Fonte:** Como o Servidor retorna somente o resultado html, o código fonte (lógica) fica preservada.

**Linguagens:** O ASP pode utilizar de comandos em VBScript, JavaScript e Html.

## Itens a verificar no seu servidor

Para utilizar ASP em suas homepages, você precisa atender aos requisitos acima. Uma situação ideal é utilizar o Microsoft FrontPage ou o Microsoft Visual InterDev para editar as páginas, mas isso não é necessário. Você pode chegar lá apenas usando o Notepad.

**Antes de mais nada:**

Certifique-se que o computador que hospedará as páginas roda Windows NT Server, Windows NT Workstation, Windows 95 ou Windows 98.

Certifique-se que você possui o Microsoft Internet Information Server (IIS), o Peer Web Services (PWS) ou o Personal Web Server (PWS, também) instalado neste computador. O Windows NT 4.0 vem com a versão 2.0 do IIS, que não é suficiente. Procure na Internet (<http://www.microsoft.com/iis>) o IIS versão 3 ou 4.

Se você pretende usar os recursos de acesso a bancos de dados, certifique-se os dados estão acessíveis através de ODBC. Você precisará de um driver de ODBC instalado e funcionando no servidor. Se você optou por usar o Access, instale uma versão 7.0 (95) ou superior no Servidor. Isto garantirá a presença do driver ODBC e facilitará a criação dos seus bancos de dados. Se você preferir utilizar o SQL Server, garanta que o driver ODBC está instalado.

Se você pretende usar os recursos de acesso a bancos de dados, verifique a existência de uma "Fonte de Dados" ODBC para este banco de dados. Isto é feito usando a opção "ODBC" do "Painel de Controle" do Windows. Se for usar o SQL Server, crie um usuário com direito para acessar este banco de dados.

Crie um diretório para guardar as páginas ASP no seu servidor. Inclua este diretório na lista do IIS/PWS, dando direito de "Execute/Execução". As páginas ASP só podem ser rodadas a partir de um diretório com o direito de "Execução". Não habilite a opção de "Read/Leitura", para aumentar sua segurança neste diretório.

Garantidos os itens acima, você pode criar sua primeira página ASP.

## Perguntas comuns sobre ASP

### **Que linguagens script são suportadas pelas ASP?**

Há suporte nativo para JScript (o JavaScript da Microsoft) e VBScript. Existem plugins ActiveX para dar suporte para outras linguagens como REXX, PERL, and Python.

### **Que browsers suportam ASP?**

Todos os browsers suportam ASP. Isto acontece pelo fato das páginas ASP serem processadas pelo servidor. O que o cliente recebe é somente código HTML padrão.

### **As páginas ASP são capazes de guardar estados?**

Sim. Aplicações ASP podem armazenar dados que são mantidos durante toda uma sessão. Desta forma, um usuário pode fornecer seu nome somente uma vez em uma página e as demais páginas podem obter este dado automaticamente. Este recurso é ideal para aplicações de venda pela Internet.

### **As páginas ASP oferecem segurança?**

Sim. O recurso ASP é parte integrante do IIS (Internet Information Server), que tem sua segurança integrada com o Windows NT Server. É fácil restringir o acesso a páginas ASP usando os esquemas de autenticação do IIS (senha básica da Web, senha do NT ou certificados de cliente). É ainda possível dar segurança aos dados transmitidos usando SSL.

### **Que bancos de dados podem ser acessados usando ASP?**

Uma aplicação ASP pode ser usada com qualquer banco de dados compatível com ODBC. Isto inclui dados do Access, Microsoft SQL Server, Oracle, Sybase, Informix, DB2, entre outros.

### **ASP é melhor que CGI?**

ASP lhe dá todos os recursos de aplicações CGI de uma forma mais fácil e mais robusta. Com ASP, é bem mais fácil criar conexões entre o browser e os dados em formatos normalmente incompatíveis com HTML, como bancos de dados. ASP é mais robusto por não criar um processo no servidor para cada pedido do usuário, como acontece com o CGI. Usando ASP ao invés de CGI, um servidor pode atender a um grande número de pedidos de usuários de forma mais rápida e usando menos memória.

Além disso, criar páginas ASP é em geral muito mais fácil do que criar aplicações CGI.

**ASP é melhor do que Perl?**

Perl é apenas uma linguagem script e não uma ferramenta de desenvolvimento. Usando ASP, você tem objetos predefinidos para criar aplicações complexas, como os que permitem o acesso a bancos de dados ou o uso de sessões. Além do mais, ASP pode utilizar Perl como linguagem script, se você desejar, usando plug-ins ActiveX de terceiros.

## TUTORIAL ASP

### **Lições básicas**

Explica os objetos fundamentais, como: response, estruturas de decisão, loops, detecção de Browser e Includes

### **Formulários**

Explica como construir formulários HTML e associá-los a scripts ASP para processá-los

### **Banco de Dados com ASP**

Tudo para combinar Bancos de Dados e páginas WEB. Fala sobre mostrar tabelas, listboxes, adicionando e atualizando registros.

### **Comandos Avançados**

Includes, subrotinas e funções, componentes content linking, componentes ad rotation, comandos de debugging e ler/gravar arquivos textos no servidor.

### **Personalizações**

Explica sobre as ferramentas (sessions, cookies e outras) para armazenar preferências dos usuários ou variáveis com outras finalidades (exemplo: referência do pagamento)

## **LIÇÕES BÁSICAS**

**IF**

**CASE**

**DO WHILE**

**FOR**

**Formatando Números**

**Formatando e manipulando datas**

**Funções para Browser**

**Response Object**

**Include Files**

## IF

Frequentemente você tem que determinar o que fazer depois que o usuário faz algum input de dados. Abaixo segue um formulário que pergunta ao usuário o seu primeiro nome e o último nome.

```
1 <html><head>
2 <TITLE>Comando IF</TITLE>
3 </head><body bgcolor="#FFFFFF">
4 <form action="ifrespond.asp" method=get>
5 Your First Name<INPUT NAME="FirstName" MaxLength=20><p>
6 Your Last Name<INPUT NAME="LastName" MaxLength=20><p>
7 <INPUT TYPE=submit><p><INPUT TYPE=reset>
8 </form></body></html>
```

Agora o arquivo ASP ifrespond.asp que examina o primeiro nome e o último nome depois que o formulário é enviado

```
1 <html><head>
2 <TITLE>ifrespond.asp</TITLE>
3 </head><body bgcolor="#FFFFFF">
4 <% fname=request.querystring("Firstname")
5 lname=request.querystring("Lastname")
6 If fname="George" and lname="Washington" then%>
7 Hi.<p>You must be the first president!
8 <%else%>
9 Hi!<p>Nice to Meet You
10 <%end if%>
11 </body></html>
```

### [Exemplo 2](#)

## IF - Exemplo 2

Abaixo temos o formulário que pergunta o primeiro nome e último nome do usuário.

```

1 <html><head>
2 <TITLE>asp_if2.htm</TITLE>
3 </head><body bgcolor="#FFFFFF">
4 <form action="if2respond.asp" method=get>
5 Your First Name<INPUT NAME="FirstName" MaxLength=20><p>
6 Your Last Name<INPUT NAME="LastName" MaxLength=20><p>
7 <INPUT TYPE=submit><p><INPUT TYPE=reset>
8 </form></body></html>

```

Agora o arquivo ASP if2respond.asp examinará o primeiro e último nome do usuário enviado pelo formulário. Ao contrário do exemplo anterior, desta vez será verificada múltiplas condições utilizando ELSEIF e formataremos as strings com letras minúsculas.

```

1 <html><head>
2 <TITLE>if2respond.asp</TITLE>
3 </head><body bgcolor="#FFFFFF">
4 <%
5 fname=lcase(request.querystring("Firstname"))
6 lname=lcase(request.querystring("Lastname"))
7 If fname="george" and lname="washington" then%>
8 Hi.<p>You must be the first president!
9 <%elseif fname="ronald" and lname="reagan" then%>
10 Hi.<p>You must be the actor president!
11 <%elseif fname="jimmy" and lname="carter" then%>
12 Hi.<p>You must be the peanut farmer president!
13 <%elseif fname="naoko" or fname="charles" then%>
14 Hi.<p>Your name reminds me of someone<p>
15 but I am not sure who!
16 <%else%>
17 Hi!<p>Nice to Meet You
18 <%end if%>
19 </body></html>

```

### [Exemplo 3](#)

## IF - Exemplo 3

Abaixo temos o formulário que pergunta o primeiro nome e último nome do usuário e salário.

```
1 <html><head>
2 <TITLE>asp_if3.asp</TITLE>
3 </head><body bgcolor="#FFFFFF">
4 <form action="if3respond.asp" method=get>
5 Your First Name<INPUT NAME="FirstName" MaxLength=20><p>
6 Your Last Name<INPUT NAME="LastName" MaxLength=20><p>
7 Your Salary <INPUT NAME="Salary" MaxLength=7><p>
8 <INPUT TYPE=submit><p><INPUT TYPE=reset>
9 </form></body></html>
```

Este exemplo mostra como o comando IF pode trabalhar com operadores

```
1 <html><head>
2 <TITLE>if3respond.asp</TITLE>
3 </head><body bgcolor="#FFFFFF">
4 <%fname=request.querystring("Firstname")
5 lname=request.querystring("Lastname")
6 salary=request.querystring("Salary")
7 response.write "Nice to Meet You " & fname & " " & lname & "<p>"
8 If salary <6700 then
9 SSTaxMarginal=0%>
10 <%If salary >6700 and salary<30000 then
11 SSTaxMarginal=0%>
12 Would you like me to loan you some money?<p>
13 <%elseif salary>100000 then%>
14 Would you like to loan me some money?<p>
15 <%sstaxMarginal=15
16 else
17 SSTaxMarginal=15
18 end if%>
19 By the way your marginal tax rate is <%=sstaxmarginal%>
20 <%end if%>
21 </body></html>
```

## SELECT CASE

Usando IF-THEN pode ser incômodo e propenso a ter erros de programação e lentidão na execução. Uma construção mais eficiente é o SELECT CASE que utiliza uma variável com várias condições.

```
1 <html><head>
2 <TITLE>asp_case.asp</TITLE>
3 </head><body bgcolor="#FFFFFF">
4 <form action="caserespond.asp" method=get>
5 Your First Name<INPUT NAME="FirstName" MaxLength=20><p>
6 Your Last Name<INPUT NAME="LastName" MaxLength=20><p>
7 <INPUT TYPE=submit><p><INPUT TYPE=reset>
8 </form></body></html>
```

Este é o select case que determinará o que significa cada input do usuário.

```
1 <html><head>
2 <TITLE>caserespond.asp</TITLE>
3 </head><body bgcolor="#FFFFFF">
4 <%
5 fname=request.querystring("Firstname")
6 lname=request.querystring("Lastname")
7 salary=request.querystring("Salary")
8 %>
9 Nice to Meet You <%=fname%> <%=lname%><p>
10 <%If fname="" then%>
11 Sorry we are not on a first name basis...<p>
12 <%end if
13 select case lcase(lname)
14 case "washington","adams"
15 response.write "The first president has same last name<p>"
16 case "jefferson"
17 response.write "The third president has same last name<p>"
18 case "lincoln"
19 response.write "The sixteenth president has same last name<p>"
20 end select%>
21 </body></html>
```

## SELECT CASE - Exemplo 2

Usando IF-THEN pode ser incômodo e propenso a ter erros de programação e lentidão na execução. Uma construção mais eficiente é o SELECT CASE que utiliza uma variável com várias condições.

```

1 <html><head>
2 <TITLE>asp_case2.asp</TITLE>
3 </head><body bgcolor="#FFFFFF">
4 <form action="case2respond.asp" method=get>
5 Your First Name<INPUT NAME="FirstName" MaxLength=20><p>
6 Your Last Name<INPUT NAME="LastName" MaxLength=20><p>
7 Your Title
8 <INPUT TYPE="Radio" name="Title" VALUE="employee">Entry Level
9 <INPUT TYPE="Radio" name="Title" VALUE="temp" CHECKED>Temporary
Employee
10 <INPUT TYPE="Radio" name="Title" VALUE="manager">Management
Candidate
11 <INPUT TYPE="Radio" name="Title" VALUE="executive">Executive
12 <INPUT TYPE="Radio" name="Title" VALUE="vice-prez">The Vice President
of...
13 <INPUT TYPE="Radio" name="Title" VALUE="CEO">The Boss<p>
14 <INPUT TYPE=submit><p><INPUT TYPE=reset>
15 </form></body></html>

```

Este é o select case que determinará o que significa cada input do usuário.

```

1 <html><head>
2 <TITLE>case2respond.asp</TITLE>
3 </head><body bgcolor="#FFFFFF">
4 <%fname=request.querystring("Firstname")
5 lname=request.querystring("Lastname")
6 title=request.querystring("title")
7 response.write "Nice to Hire You " & fname & " " & lname & "<p>"
8 Select Case lcase(Title)
9 case "employee", "temp"
10 response.write("The washroom is in the hall")
11 case "manager", "executive"
12 response.write("Here is your key to the Executive washroom")
13 case "CEO", "vice-prez"
14 response.write("The maid will attend to your private washroom")
15 End Select%>
16 </body></html>

```

## DO WHILE

Para revisar, quando falamos de estruturas de controle significa instruções que fazem com que o programa rode em outra ordem que não seja a linha 1, linha 2, linha 3 etc. IF...Then e Select Case fazem com que linhas do código sejam executadas e outras não. Call e GoTo fazem com que o programa vá para outra localização no código. As estruturas de **Loop** fazem com que alguns comandos sejam repetidos. As estruturas de Loop possuem duas opções: For...Next e Do While ... Loop. Se você conhecer a quantidade de repetições que devem ser realizadas, utilize o comando FOR...NEXT, caso não saiba, utilize a estrutura Do While...Loop.

Todas as linhas de código dentro dos comandos DO WHILE e LOOP serão repetidas várias vezes até que o teste que vem depois do WHILE seja falso. Assim as repetições param e o programa passa para o código seguinte ao LOOP.

Sintaxe:

DO WHILE condição

linhas de código

LOOP

Existe a possibilidade do Loop nunca acabar. Então se previna usando um IF para terminar, como:

```
contador=1
```

```
contador = contador +1
```

```
If contador>100 then
```

```
exit do
```

```
End If
```

### Exemplo 1:

```
1 <html><head>
2 <title>DoLoop2.asp</title>
3 <body bgcolor="#FFFFFF"></head>
4 <p>DoLoop2.asp Exemplo<br>
5 Faz com que o programa imprima somente a quantidade de meses.</p>
6 <p>
7 <P>
8 <%
9 contador = 1
10 mes_atual = month(now())
11 Do while contador < mes_atual + 1
12 response.write "Número do Mês = " & contador & " "
13 response.write "_____ " & "<BR><br>"
14 If contador >13 then
15 exit do
16 end if
17 contador = contador+1
18 Loop
19 %>
20 <hr></body></html>
```

**Exemplo 2:**

```
1 <html><head>
2 <title>DoLoop3.asp</title>
3 </head><body bgcolor="#FFFFFF">
4 <form action="DoLoopBack.asp" method = post>
5 <p>DoLoop3 Exemplo<br>
6 instead of form to print, make form for electronic submission</p>
7 <p></p>
8 <%
9 contador = 1
10 mes_atual = month(now())
11 Do while contador < mes_atual + 1
12 response.write "Número do mês " & contador & " "
13 TempTextField = "<input type = " & chr(34) & "text" & chr(34)
14 TempTextField = TempTextField & "name=" & chr(34) & contador & chr(34)
15 TempTextField = TempTextField & ">"
16 response.write TempTextField & "<br><br>"
17 If contador >13 then
18 exit do
19 end if
20 contador = contador+1
21 Loop
22 %>
23 <input type=submit>
24 </form><hr></body></html>
```

## FOR NEXT

O Loop com o número inicial e vai até o número final. Como foi colocado anteriormente, o FOR...NEXT é para ser usado quando se sabe a quantidade de repetições que devem ser feitas.

Sintaxe:

```
For contador = inicio to fim  
  linhas de código
```

Next

Para que seu loop não seja infinito você pode colocar um IF para pará-lo, como:

```
If contador>100 then  
  exit For  
End If
```

### Exemplo:

```
1 <html><head>  
2 <title>fornext.asp</title>  
3 <body bgcolor="#FFFFFF"></head>  
4  
5 exemplo<br>Espere e veja o funcionamento do for...next<br>  
6 <%for contador = 1 to 5  
7 response.write "estamos no looping" & "<br>"  
8 next%><hr>  
9  
10 exemplo 2<br>Você pode usar a variável contador no seu código<br>  
11 <%for contador = 1 to 5  
12 response.write "Agora o número de loops aparecem " & contador & "<br>"  
13 next%><hr>  
14  
15 exemplo 3<br>Você pode dar passos maiores que 1<br>  
16 <%for contador = 0 to 25 step 5  
17 response.write "Agora o número do loop é: " & contador & "<br>"  
18 next%><hr>  
19  
20 exemplo 4<br>Você não precisa começar do 1<br>  
21 <%for contador = 25 to 50 step 5  
22 response.write "Agora o número do loop é: " & contador & "<br>"  
23 next%><hr>  
24  
25 exemplo 5<br>  
26 (mude os passos para negativo e tenha certeza do início ser maior que o fim)<br>  
27 <%for contador = 50 to 25 step -5  
28 response.write "Agora o número do loop é: " & contador & "<br>"  
29 next%><hr>  
30 </body></html>
```

## FORMATNUMBER

Freqüentemente você quer que o número apareça em um determinado formato. A função `FormatNumber` formata o número e retorna em um específico formato.

Sintaxe: `FormatNumber(expression, iDigits, bleadingDigit, bParen, bGroupDigits)`

ARGUMENTO	SIGNIFICADO
Expression	variável que contém o número
IDigits	Número de dígitos depois da casa decimal
BleadingDigit	1 para mostrar os zeros antes da vírgula 0 para não mostrar os zeros antes da vírgula
bParen	1 para parêntesis ao redor de números negativos 0 para não ter parêntesis ao redor de números negativos
BGroupDigits	1 para mostrar o número de acordo com as Configurações Regionais do Painel de Controle 0 para ignorar as configurações do Painel de Controle

### Exemplo:

```

1 <html><head>
2 <TITLE>formatnumbers.asp</TITLE>
3 </head><body bgcolor="#FFFFFF">
4 <%
5 ' Exemplo de formatação de números
6 mynumber=123.4567
7 response.write "<hr>" & mynumber & "<br>"
8 response.write "formatnumber(mynumber,0)" & "<br>"
9 response.write formatnumber(mynumber,0) & "<hr>"
10 response.write "formatnumber(mynumber,2)" & "<br>"
11 response.write formatnumber(mynumber,2) & "<hr>"
12 response.write "formatnumber(mynumber,6)" & "<br>"
13 response.write formatnumber(mynumber,6) & "<hr>"
14 mynumber=.4567
15 response.write mynumber & "<br>"
16 '0 significa não mostrar os zeros antes da virgula
17 response.write "formatnumber(mynumber,2,0)" & "<br>"
18 response.write formatnumber(mynumber,2,0) & "<hr>"
19 '1 significa mostrar os zeros antes da virgula
20 'response.write "formatnumber(mynumber,2,1)" & "<br>"
21 'response.write formatnumber(mynumber,2,1) & "<hr>"
22 'mynumber=-123.4567
23 'response.write mynumber & "<br>"
24 '0 não mostra os parêntesis para números negativos
25 'response.write "formatnumber(mynumber,2,0,0)" & "<br>"
26 'response.write formatnumber(mynumber,2,0,0) & "<hr>"
27 '1 mostra os parêntesis para números negativos
28 'response.write "formatnumber(mynumber,2,0,1)" & "<br>"
29 'response.write formatnumber(mynumber,2,0,1) & "<hr>"
30 %>
31 </body></html>

```

## FORMATANDO DATAS

A maneira mais fácil de demonstrar a formatação de datas é mostrar um código exemplo com várias permutações do comando.

**FormatDateTime**(Data[,Formato\_do\_nome])

A função **FormatDateTime** tem as seguintes partes:

PARTE	DESCRIÇÃO
Data	Obrigatório. Data que será formatada
Formato_do_nome	Opcional. Valor numérico que indica o formato date/time que será usado. Se for omitido, o formato <b>vbGeneralDate</b> será usado.

### Parâmetros

O argumento Formato\_do\_nome tem o seguinte domínio:

CONSTANTE	VALOR	DESCRIÇÃO
<b>VbGeneralDate</b>	<b>0</b>	Mostra a data e/ou hora. Se houver uma data, mostrará como short date. Se for hora, mostrará como long time. Se houver data e hora ambas serão mostradas.
<b>VbLongDate</b>	<b>1</b>	Mostra a data usando o formato long date especificado nas Configurações Regionais do Painel de Controle
<b>VbShortDate</b>	<b>2</b>	Mostra a data usando o formato short date especificado nas Configurações Regionais do Painel de Controle.
<b>VbLongTime</b>	<b>3</b>	Mostra a hora usando o formato especificado nas Configurações Regionais do Painel de Controle.
<b>VbShortTime</b>	<b>4</b>	Mostra a hora usando o formato 24-horas (hh:mm).

```

1 <html><head><title>formatdates.asp</title></head><body bgcolor="#FFFFFF">
2 <% 'ASP que formata data
3 response.write "<hr>"
4 for counter=0 to 4
5 currentdate=now()
6 response.write "today is..." & "<br>"
7 response.write currentdate & "<P>"
8 select case counter
9 case 0
10 whichformat="vbgeneraldate"
11 case 1
12 whichformat="vblongdate"
13 case 2
14 whichformat="vbshortdate"
15 case 3
16 whichformat="vblongtime"
17 case 4
18 whichformat="vbshorttime"
19 end select
20 response.write "FormatDate(now()," & whichformat & ")="
21 response.write Formatdatetime(currentdate,counter) & "<P><HR>"
22 next%>
23 </body></html>

```

## FUNÇÕES DE BROWSER

### (Browser Capabilities)

O script abaixo demonstra a maioria das propriedades usadas do componente **Browser Capabilities**.

```
1 <html><head>
2 <TITLE>asp_browser.htm</TITLE>
3 </head><body bgcolor="#FFFFFF">
4 <% Set bc = Server.CreateObject("MSWC.BrowserType") %>
5 Browser Name: <%= bc.browser %><p>
6 Browser Version: <%= bc.version %><p>
7 <% if (bc.frames = TRUE) then %>
9 Você pode usar frames<p>
10 <% else %>
11 Você não pode usar frames<p>
12 <% end if %>
13 <% if (bc.tables = TRUE) then %>
14 Você pode usar tabelas<p>
15 <% else %>
16 Você não pode usar tabelas<p>
17 <% end if %>
18 <% if (bc.BackgroundSounds = TRUE)then %>
19 Você me permite tocar músicas<p>
20 <% else %>
21 Você não colocou músicas no código<p>
22 <% end if %>
23 <% if (bc.vbscript = TRUE) then %>
25 Você pode rodar script da linguagem VBscript<p>
26 <% else %>
27 Você não pode rodar script da linguagem VBscript<p>
28 <% end if %>
29 <% if (bc.javascript = TRUE) then %>
31 Você pode rodar script da linguagem JScript<p>
32 <% else %>
33 Você não pode rodar script da linguagem JScript<p>
34 <% end if %>
35 </BODY>
36 </HTML>
```

## RESPONSE OBJECT

Existem várias combinações do objeto Response que não serão vistas aqui, mas tenho certeza que estes 20% que será visto será 80% do que você vai utilizar. As propriedades vitais são:

```
response.write
response.redirect
response.end
```

Algumas notas sobre response.write, & e =	Exemplos adicionais com &
<p>Os próximos 3 exemplos têm a mesma saída.</p> <p><b>Exemplo 1a:</b></p> <pre>&lt;%   response.write "Livro? &lt;input type=" &amp; "" &amp; "TEXT" &amp; "" &amp; "name=" &amp; "" &amp; "livro" &amp; "" &amp; "value=" &amp; "" &amp; livro &amp; "" &amp; "&gt;&lt;/p&gt;" %&gt;</pre> <p><b>Exemplo 2a (mais legível):</b></p> <pre>livro? &lt;input type="TEXT" name="livro" value="&lt;%=livro%&gt;"&gt;&lt;/p&gt;</pre> <p><b>Exemplo 3a:</b></p> <pre>livro? &lt;input type="TEXT" name="livro" value=" &lt;%response.write livro %&gt; "&gt;&lt;/p&gt;</pre>	<p>Os próximos 3 exemplos têm a mesma saída.</p> <p><b>Exemplo 1b: Envia "Oi José&lt;p&gt;" para o browser</b></p> <pre>&lt;%   x="Oi "   x = x &amp; "José"   response.write x &amp; "&lt;p&gt;" %&gt;</pre> <p><b>Exemplo 2b: Envia "Oi José&lt;p&gt;" para o browser</b></p> <pre>&lt;% x="Jose &lt;p&gt;"   response.write "Oi " &amp; x &amp; "&lt;p&gt;" %&gt;</pre> <p><b>Exemplo 3b: Envia "Oi José&lt;p&gt;" para o browser</b></p> <pre>&lt;% x ="José" %&gt; Oi &lt;%=x%&gt;&lt;p&gt;</pre>

### Script utilizando o response.write:

```
1 <%
2 'response.buffer=true
3 response.buffer=false
4 %>
5 <html>
6 <head>
7 <title>res.asp</title>
8 </head>
9 <body bgcolor="#FFFFFF">
10 <%
11 ' Este programa demonstra as funções básicas do response
12 response.write "Oi, How Are You Doing"
13 for counter=1 to 5000
14 response.write counter & ", "
15 next
16 %>
17 </body></html>
```

[Segunda Parte](#)

## RESPONSE OBJECT

### Segunda Parte - Propriedade Buffer

```
<%response.buffer=true%>
```

Toda página desenvolvida com o FrontPage ou qualquer outro editor de HTML que misturam headers (objetos ASP de servidor) e/ou texto deverá conter a linha acima antes. Se não for colocada, geralmente aparecerá uma mensagem semelhante a "headers are already sent". Esse erro costuma aparecer quando você precisa usar o propriedade response.redirect, pois essa altera o header do script. Este header não pode ser mudado se você já descarregou o buffer.

```
<%response.buffer=true%>
```

Essencialmente diz para o browser não gravar nada até que:

**a)** response.end seja executado parando a execução e enviando as informações gravadas.

**b)** response.flush seja executado descarregando o buffer.

**c)** 100% da página é enviada para o browser. Se o comando response.redirect for enviado e existir o comando response.buffer=true o texto só será enviado para o browser se ocorrer **a)** ou **b)**.

Se você estiver em um While que trata milhares de registros em um banco de dados e você ajustou a propriedade buffer para true, a página não aparecerá até que todos os dados estejam no buffer para serem enviados ao browser cliente. Para resolver esse inconveniente, basta usar ocasionalmente o response.flush para que as porções gravadas no buffer sejam descarregadas enquanto o servidor termina de construir o restante da página.

### Terceira Parte

## RESPONSE OBJECT

### Parte 3 - Redirection Code

O objeto response.redirect pode ser usado para direcionar o usuário para uma determinada página. No exemplo abaixo cada link se refere ao mesmo arquivo ASP, entretanto, com o parâmetro **where** o programa redireciona o usuário para um determinado lugar.

[res2.asp?where=principal](#)

[res2.asp?where=exemplos](#)

[res2.asp?where=docs](#)

[res2.asp?where=diversao](#)

[res2.asp?where=news](#)

[res2.asp](#)

```
1 <%
2 response.buffer=true
3 %>
4 <html><head>
5 <title>res2.asp</title></head>
6 <body bgcolor="#FFFFFF">
7 <%
8 ' programa que redireciona URL
9 URL_atual = "http://www.geap.com.br"
10 where=Request.QueryString("Where")
11 Select Case where
12 case "principal"
13 response.redirect URL_atual & "\"
14 case "exemplos"
15 response.redirect URL_atual & "/exemplos/exemplos.htm"
16 case "docs"
17 response.redirect URL_atual & "/aspdocs/roadmap.asp"
18 case "news"
19 response.redirect "http://www.cnn.com"
20 case "diversao"
21 response.redirect "http://www.dilbert.com"
22 End Select
23 response.write "Não sei para onde você foi, mas..."
24 response.write "Eu recomendo --> " & "<P>"
25 response.write server.htmlencode(URL_atual & "/pagamento") & "<P>"
26 response.write "para ver vários exemplos de ASP!" & "<P>"
27 %>
28 </body></html>
```

## INCLUDE FILES

O recurso Include Files é uma maneira de tornar o código ASP eficiente e reaproveitar rotinas já prontas. No arquivo Include devem constar somente funções e procedures que podem ser usadas por um ou muitos sistemas. Basicamente existem duas formas de declará-lo:

```
<!--#include virtual="meu_arquivo.asp"-->
<!--#include file="meu_arquivo.asp"-->
```

Muitos desenvolvedores usam a extensão .inc, mas qualquer extensão pode ser usada.

**IMPORTANTE:** Os Include files são sempre processados e inseridos no programa antes de qualquer ASP Script. Preferencialmente no início do arquivo.

## FORMULÁRIOS

Formulários HTML são geralmente o "front end" de um script ASP. A divisão abaixo facilitará o aprendizado.

### **Introdução sobre Formulários**

#### **Text Box**

#### **Text Area**

#### **Check Box**

#### **Radio Buttons**

#### **List Box**

## INTRODUÇÃO SOBRE FORMULÁRIOS

Formulários são o caminho natural para os usuários enviarem informações para o ASP.

Formulários podem ser HTML ou ASP dependendo das capacidades dinâmicas que você quer.

Formulário deve passar variáveis para o arquivo ASP para processar os dados.

Você pode encontrar um excelente tutorial sobre formulários em <http://www.mountaindragon.com/html/forms.htm> (tutorial em inglês)

### Formulário com GET

```
<form action="x.asp" method=get>
...
<input type=submit><input type=reset>
</form>
```

O formulário pode mostrar as informações dos campos na tela do browser.

Um script ASP pode pegar os dados dos campos do formulário da seguinte forma:  
<%variável=request.querystring("nome\_do\_campo")%>

### Formulário com POST

```
<form action="x.asp" method=post>
...
<input type=submit><input type=reset>
</form>
```

O formulário não mostrará as informações na tela do browser. Ele enviará para o arquivo especificado no action do form e este arquivo fará o tratamento das informações.

Um script ASP pode pegar os dados dos campos do formulário da seguinte forma:  
<%variável=request.form("nome\_do\_campo")%>

## FORMULÁRIOS - TEXT BOX

```
<INPUT NAME="UltimoNome">
```

Este comando criará um input box com o tamanho default e o browser passará para o arquivo output em ASP a variável com o nome: UltimoNome.

```
<INPUT NAME="CEP" SIZE="10">
```

Este parâmetro não limita o número de caracteres que podem ser digitados. Não use esse parâmetro como técnica de validação. Ele apenas define o tamanho da caixa de texto.

```
<INPUT NAME="Estado"
MaxLength="2">
```

Agora sim. Esse parâmetro define o tamanho máximo de caracteres digitados.

```
<INPUT NAME="UltimoNome"
VALUE="Bertrand">
```

O nome Bertrand aparecerá no campo texto quando a página for carregada. É um valor default.

```
1 <html><head>
2 <title>FormTextBox.asp</title>
3 </head><body bgcolor="#FFFFFF">
4 <Form action = "FormTextBoxRespond.asp" method=GET>
5 Fill Out This Form For Us:<p>
6 Last Name -> <Input NAME="UltimoNome" size ="10"><br>
7 País -> <Input NAME="País" value="USA" size=10><br>
8 Estado -> <Input NAME="Estado" MaxLength="2" size=2><br>
9 <Input type=submit><Input type=reset>
10 <hr></form>
11 </body></html>
```

```
1 <html><head>
2 <title>FormTextBoxRespond.asp</title>
3 </head><body bgcolor="#FFFFFF">
4 <%Iname=request.querystring("UltimoNome")
5 cty=request.querystring("País")
6 st=request.querystring("Estado")
7 response.write Iname & "<br>"
8 response.write city & "<br>"
9 response.write st & "<br>"%>
10 </body></html>
```

## FORMULÁRIOS - TEXTAREA

O comando TEXTAREA cria uma caixa de texto com múltiplas linhas, conforme segue:

```
<TEXTAREA NAME="Comentarios" ROWS=5 COLS=50>
... o texto padrão é digitado aqui
</TEXTAREA>
```

Nota: O browser geralmente coloca a scrollbars automaticamente.

```
1 <html><head>
2 <TITLE>textarea.asp</TITLE>
3 </head><body bgcolor="#FFFFFF">
4 <hr><form action="FormTextAreaRespond.asp" method=post>
5 <p>TextArea Exemplo</p>
6 <p>Por Favor, coloque seu comentário:</p>
7 <TEXTAREA NAME="ComentarioDigitado" ROWS=5 COLS=50>
8 comentarios aparecerão aqui.
9 </textarea>
10 <p><input type=submit><input type=reset>
11 </form><hr>
12 </body></html>
```

**A resposta para o formulário será semelhante a esta:**

```
1 <html><head>
2 <TITLE>textarearespond.asp</TITLE>
3 </head><body bgcolor="#FFFFFF">
4 <%
5 comm=request.form("ComentarioDigitado")
6 response.write comm
7 %>
8 </form><hr>
9 </body></html>
```

## FORMULÁRIOS - CHECK BOXES

O objeto checkbox tem as mesmas linhas de código dos radio buttons, entretanto cada checkbox deve ter um único nome porque o estado de "checked" ou "not checked" será passado para o script ASP. Lembre-se que você pode marcar mais de uma opção.

```

1 <html><head>
2 <TITLE>FormCheckBox.asp</TITLE>
3 </head><body bgcolor="#FFFFFF">
4 <form action="FormCheckBoxRespond.asp" method="post">
5 <p>CheckBox Exemplo</p>
6 <p>Como você quer comprar?</p>
7 <input TYPE="checkbox" NAME="Correio">Confirmação será pelo Correio<br>
8 <input TYPE="checkbox" NAME="Sedex">Confirmação enviada via Sedex<br>
9 <input TYPE="checkbox" NAME="EMail" CHECKED>Confirmação por
EMail<br>
10 <input TYPE="checkbox" NAME="Fax">Confirmação enviada por Fax<br>
11 <input TYPE="checkbox" NAME="Tel" CHECKED>Confirmação feita por
telefone<br><br>
12 <input type="submit"><input type="reset">
13 </form><hr></body></html>

```

#### A resposta será essa:

```

1 <html><head>
2 <TITLE>formCheckBoxRespond.asp</TITLE>
3 </head><body bgcolor="#FFFFFF">
4 <%
5 If request.form("Correio")="on" then
6 response.write "<br>Nós confirmaremos por Correio"
7 end if
8 If request.form("Sedex")="on" then
9 response.write "<br>Nós confirmaremos por Sedex"
10 end if
11 If request.form("EMail")="on" then
12 response.write "<br>Nós confirmaremos por EMail"
13 end if
14 If request.form("fax")="on" then
15 response.write "<br>Nós confirmaremos por fax"
16 end if
17 If request.form("tel")="on" then
18 response.write "<br>Nós confirmaremos por tel"
19 end if%>
20 </body></html>

```

## FORMULÁRIOS - RADIO BUTTONS

Todos os inputs no formulário devem ter seu nome único que o identifica para uso do script ASP. Os Radio Buttons são uma exceção. Entre as opções dos Radio Buttons, somente um valor será passado para o script ASP, isto implica que todas as opções tenham o mesmo nome. O botão selecionado como default deve ser indicado com o comando CHECKED. Isso é importante porque o browser enviará ao ASP um nome e um valor referente ao botão selecionado pelo usuário. O browser não enviará para o ASP o texto que está associado com o botão. Como no exemplo abaixo, se o usuário checar o botão com o texto SALVADOR, o ASP receberá Cidade = SSA

```

1 <html><head>
2 <TITLE>formRadio.asp</TITLE>
3 </head><body bgcolor="#FFFFFF">
4 <form action="FormRadiorespond.asp" method="post">
5 <p>Radio Buttons </p><br><p>Exemplo 1</p>
6 <p>Qual regional você gostaria de visitar?</p>
7 <input TYPE="radio" NAME="Cidade" VALUE="SSA">Salvador
8 <input TYPE="radio" NAME="Cidade" VALUE="BSB">Brasilia
9 <input TYPE="radio" NAME="Cidade" VALUE="SPA" CHECKED>São Paulo
10 <br><br><input type="submit"><input type="reset">
11 </form><hr>
12 </body></html>

```

A resposta se parecerá com:

```

1 <html><head>
2 <TITLE>formradiorespond.asp</TITLE>
3 </head><body bgcolor="#FFFFFF">
4 <%minhaCidade = request.form("Cidade")
5 Select Case ucase(MinhaCidade)
6 case "SSA"
7 response.write "Salvador te espera no Carnaval"
8 case "BSB"
9 response.write "Brasilia te aguarda para o Impeachment do FHC"
10 case "SPA"
11 response.write "São Paulo é a cidade mais poluída, mas pode vir assim mesmo."
12 End Select%>
13 </body>
14 </html>

```

## FORMULÁRIOS - LIST BOXES

O objeto listbox é muito usado para facilitar a entrada do usuário. É apenas um lista pulldown.

```

1 <html><head>
2 <TITLE>FormListBox.asp</TITLE>
3 </head><body bgcolor="#FFFFFF">
4 <form action="FormListBoxRespond.asp" method="post">
5 <SELECT NAME="Estado">
6 <OPTION SELECTED VALUE="DF">Distrito Federal</OPTION>
7 <OPTION value="BA">Salvador</OPTION>
8 <OPTION value="RJ">Rio de Janeiro</OPTION>
9 <OPTION value="outros">O Resto!</OPTION>
10 <input type=submit>
11 </form><hr></body></html>

```

A resposta se parecerá com:

```
1 <html><head>
2 <TITLE>formlistboxrespond.asp</TITLE>
3 </head><body bgcolor="#FFFFFF">
4 <%meuEstado = request.form("Estado")
5 Select Case ucase(meuEstado)
6 case "DF"
7 response.write "Brasilia te espera no final do mês."
8 case "BA"
9 response.write "O Carnaval está chegando..."
10 case else
11 response.write "nenhum compromisso agendado..."
12 End Select%>
13 </body>
14 </html>
```

## BANCO DE DADOS NA WEB

Banco de Dados é a razão para muitos desenvolvedores utilizar o ASP e abaixo está colocado exemplos e formatos típicos para manipular dados.

### **Abrindo Banco de Dados**

Via DSN e sem DSN

### **Mostrando tabelas**

Através de consultas SQL

### **List Boxes**

Preenchidas através de banco de dados

### **Adicionando Registros**

Em um banco de dados com o comando SQL insert Into

### **Adicionando Registros**

Em um banco de dados com o método ADO add new

### **Exemplo Completo**

Inclui o display de tabelas vinculadas a um formulário que rodam scripts que atualizam registros.

### **Atualizando registros**

Em um banco de dados.

## ABRINDO BANCO DE DADOS

Qualquer script ASP que precisa conectar em um banco de dados, deve primeiramente abri-lo no servidor. Para isso existem duas formas:

**uma conexão via DSN**

**uma conexão sem DSN**

Uma conexão via DSN requer que o Administrador do banco ou da rede, configure um DSN no servidor Internet/Intranet usando o Painel de Controle (ODBC 32 bits) ou utilize um componente ASP que implementado no seu script pode fazer mudanças na registry do servidor e criar o DSN necessário. Esses componentes são de terceiros e podem ser encontrados em sites especializados em ASP.

Conexões via DSN geralmente requerem um nome de DSN, um usuário e uma senha.

Abaixo temos o código que abre um banco com DSN igual a "estudante", usuário igual a "estudante" e uma senha igual a "magic". Isso segue o mesmo procedimento realizado nos relatórios do sistema de pagamento.

```
1 set conntemp=server.createobject("adodb.connection")
2 conntemp.open "estudante","estudante","magic"
3 set rstemp=conntemp.execute("select * from autor")
```

O que acontece se não tivermos um DSN? Se você conhece o nome do arquivo (baseado em databases como Access, Paradox, FoxPro, etc.) ou o nome do Data Source (SQLserver por exemplo) tudo pode ser resolvido. Abaixo está uma maneira de abrir um Data Source sem um DSN. Note que você deve conhecer o path completo do arquivo no servidor, isto é, msapgt.mdb não é suficiente. É preciso ter "Z:\users\pagamento\msapgt.mdb". Existe uma função no ASP chamada **server.mappath** que pega um nome de arquivo e retorna o path completo do arquivo no servidor, mas não é muito segura.

```
1 set conntemp=server.createobject("adodb.connection")
2 cnpath="DBQ=" & server.mappath("sua_tabela.mdb")
3 conntemp.Open "DRIVER={Microsoft Access Driver (*.mdb)}; " & cnpath
4 set rstemp=conntemp.execute("select * from autor")
```

Aqui estão os tipos mais comuns de nomes para drives ODBC que você pode precisar:

Para Access --> driver = {Microsoft Access Driver (\*.mdb)};

Para SQL -----> driver = SQL Server;

## TABELA COM CONSULTA SQL

Esta página demonstra a utilização de tabela construída a partir de uma consulta SQL.

```
1 <TITLE>dbtable.asp</TITLE>
2 <body bgcolor="#FFFFFF">
3 <%
4 ' Programa ASP que se comunica com um banco de dados
5 set conntemp=server.createobject("adodb.connection")
6 ' DSN, usuário e senha
7 conntemp.open "Student","student","magic"
8 set rstemp=conntemp.execute("select * from authors where AU_ID<100")
9 qtde_campos=rstemp.fields.count -1
10 %>
11 <table border=1>
12 <tr>
13 <% 'Coloca o cabeçalho de cada coluna com o nome do campo
14 for i=0 to qtde_campos %>
15 <td><b><%=rstemp(i).name %></B></TD>
16 <% next %>
17 </tr>
18 <% ' Preenche a tabela com os registros
19 do while not rstemp.eof %>
20 <tr>
21 <% for i = 0 to qtde_campos%>
22 <td valign=top><%= rstemp.fields(i).value %></td>
23 <% next %>
24 </tr>
25 <% rstemp.movenext
26 loop
27 conntemp.close%>
28 </table>
29 </BODY>
30 </HTML>
```

## LIST BOXES

Esta página mostra a funcionalidade de mostrar uma list box através de uma consulta SQL. O exemplo abaixo é o mais simples possível. Existem outros tipos utilizando Múltiplas List Boxes e Modularidade. Os exemplos utilizam um DSN, mas você pode executá-los sem.

```
1 <html><head>
2 <TITLE>dblist.asp</TITLE>
3 </head><body bgcolor="#FFFFFF">
4 <%
5 ' Conexão com o banco de dados
6 set conntemp=server.createobject("adodb.connection")
7 ' DSN usuário senha
8 conntemp.open "Estudante","Estudante","magic"
9 set rstemp=conntemp.execute("select autor from autores where AU_ID<100")
10 %>
11 <Form>
12 <Select>
13 <% ' Loop para preencher os nomes
14 do while not rstemp.eof %>
15 <option> <%=RStemp(0)%> </option>
16 <% rstemp.movenext
17 loop
18 conntemp.close%>
19 </Select>
20 </form>
21 </BODY>
22 </HTML>
```

## ADICIONANDO REGISTROS COM STATEMENT SQL

Esta página demonstra como adicionar registros em um banco de dados com um statement sql. Para melhor entender o exemplo, suponha que a página chamada seja: [http://www.geap.com.br/tutorial\\_asp/asp\\_dbnewSQL.asp?nome="Alexandre Barreto"&id=9000&ano=1974](http://www.geap.com.br/tutorial_asp/asp_dbnewSQL.asp?nome=)

```
1 <TITLE>dbnewrecSQL.asp</TITLE>
2 <body bgcolor="#FFFFFF">
3 <HTML>
4 <%
5 on error resume next
6 aunome=request.querystring("nome")
7 auano=request.querystring("ano")
8 auID=request.querystring("ID")
9 Set Conn = Server.CreateObject("ADODB.Connection")
10 ' DSN usuário senha
11 conn.open "Estudante","Estudante","magic"
12 SQLStmt = "INSERT INTO autores (AU_ID,autor,ano_nasc) "
13 SQLStmt = SQLStmt & "VALUES (" & auid
14 SQLStmt = SQLStmt & "," & aunome & ""
15 SQLStmt = SQLStmt & "," & int(auano) & ")"
16 Set RS = Conn.Execute(SQLStmt)
17 If err.number>0 then
18 response.write "Ocorreram Erros:" & "<P>"
19 response.write "Erro Número =" & err.number & "<P>"
20 response.write "Descrição =" & err.description & "<P>"
21 response.write "Help Context =" & err.helpcontext & "<P>"
22 response.write "Help Path =" & err.helppath & "<P>"
23 response.write "Native Error =" & err.nativeerror & "<P>"
24 response.write "Source =" & err.source & "<P>"
25 response.write "SQLState =" & err.sqlstate & "<P>"
26 else
27 response.write "Não ocorreram erros!!" & "<P>"
28 end if
29 IF conn.errors.count> 0 then
30 response.write "Ocorreram erros no banco de dados" & "<P>"
31 for counter = 0 to conn.errors.count
32 response.write "Número " & conn.errors(counter).number & "<P>"
33 response.write "Descrição -> " & conn.errors(counter).description & "<P>"
34 next
35 else
36 response.write "Não ocorreram erros no banco de dados !" & "<P>"
37 end if
38 Conn.Close
39 %>
40 </BODY></HTML>
```

## ADICIONANDO REGISTROS COM O MÉTODO ADO ADD NEW

```
1 <TITLE>dbnewADO.asp</TITLE>
2 <body bgcolor="#FFFFFF">
3 <HTML>
4 <%
5 on error resume next
6 auname=request.querystring("nome")
7 auano=request.querystring("ano")
8 auID=request.querystring("ID")
9 Set Conn = Server.CreateObject("ADODB.Connection")
10 ' DSN usuário senha
11 conn.open "Estudante","Estudante","magic"
12 Set RS = Server.CreateObject("ADODB.Recordset")
13 RS.Open "autores",Conn,3,3
14 RS.AddNew
15 RS("AU_ID")=auID
16 RS("autor") = auname
17 RS("ano_nasc")= int(auano)
18 RS.Update
19 RS.Close
20 Conn.Close
21 If err.number>0 then
22 response.write "Ocorreram Erros:" & "<P>"
23 response.write "Número=" & err.number & "<P>"
24 response.write "Descrição =" & err.description & "<P>"
25 response.write "Help Context =" & err.helpcontext & "<P>"
26 response.write "Help Path =" & err.helppath & "<P>"
27 response.write "Native Error =" & err.nativeerror & "<P>"
28 response.write "Source =" & err.source & "<P>"
29 response.write "SQLState =" & err.sqlstate & "<P>"
30 else
31 response.write "Nenhum erro apareceu!" & "<P>"
32 end if
33 IF conn.errors.count> 0 then
34 response.write "Ocorreram erros no banco de dados !" & "<P>"
35 for counter = 0 to conn.errors.count
36 response.write "Número " & conn.errors(counter).number & "<P>"
37 response.write "Descrição -> " & conn.errors(counter).description & "<P>"
38 next
39 else
40 response.write "Nenhum erro ocorreu no banco de dados !" & "<P>"
41 end if
42 %>
43 </BODY></HTML>
```

## EXEMPLO COMPLETO

### 1a. Parte

Vamos analisar 3 exemplos de script.

O primeiro exemplo (1a. parte) mostra uma tabela parecida com o exemplo de tabelas que foi visto anteriormente, com exceção dos links que aparecerão nas colunas iniciais. Clicando em um dos links o segundo exemplo será chamado e receberá um ID como parâmetro. No segundo exemplo será permitido alterar as informações e após clicar no botão Submit, o exemplo 3 será chamado para processar as informações e atualizá-las no banco em SQL. Vamos por partes...

### Exemplo 1

```

1 <html>
2 <head>
3 <title>dbfull1.asp</title>
4 </head>
5 <body bgcolor="#FFFFFF">
6 <body bgcolor="#FFFFFF">
7 <%
8 ' Conexão com o banco de dados
9 set conntemp=server.createobject("adodb.connection")
10 conntemp.open "Estudante","Estudante","magic"
11
12 set rstemp=conntemp.execute("select * from autores where AU_ID<100")
13 qtde_campos=rstemp.fields.count - 1
14 %>
15 <table border="1">
16 <tr>
17 <td valign="top">---</td>
18 <% 'Preenche a primeira linha com o nome dos campos
19 for i=0 to qtde_campos %>
20 <td><b><%=rstemp(i).name %></b></td>
21 <% next %>
22 </tr>
23 <% ' Preenche a tabela com os registros do banco de dados
24 do while not rstemp.eof %>
25 <tr>
26 <td valign="top"><a
27 HREF="dbfull2.asp?str_ID=<%=rstemp("AU_ID")%>">Editar</a></td>
28 <td valign="top"><%= rstemp.fields(i).value %></td>
29 <% next %>
30 </tr>
31 <% rstemp.movenext
32 loop
33 conntemp.close%>
34 </table>
35 </body>
36 </html>

```

**Exemplo 2 (2a. Parte)**

```
1 <html><head>
3 <title>dbfull2.asp</title>
4 </head>
5 <body bgcolor="#FFFFFF">
6 </body>
7 </html>
8 <html>
9 <% ' Conecta com o banco e pega o valor do str_ID permitindo a edição do registro
10 set conntemp=server.createobject("adodb.connection")
11 conntemp.open "Estudante","Estudante","magic"
12
13 ID=request.querystring("str_ID")
14 sqltemp="select * from autores where AU_ID=" & ID
15 set rstemp=conntemp.execute(sqltemp)
16 atual_ID=rstemp("AU_ID")
17 atual_nome=rstemp("autor")
18 atual_ano_nasc=rstemp("ano_nasc")
19 %>
20 <body>
21 <form name="meu_autor" action="dbfull3.asp" method="GET">
22 <p>ID: <input type="TEXT" name="id" value="<%=atual_id%>"></p>
23 <p> Nome do autor: <input type="TEXT" name="nome"
value="<%=atual_nome%>"></p>
24 <p> Ano de Nascimento: <input type="TEXT" name="ano"
value="<%=atual_ano_nasc%>"></p>
25 <p> <input type="SUBMIT"> </p>
26 </form></body>
```

**Exemplo 3 (3a. Parte)**

```
1 <HTML><HEAD>
2 <TITLE>dbfull3.asp</TITLE>
3 <body bgcolor="#FFFFFF"></HEAD>
4 <%
5 on error resume next
6 au_nome=request.querystring("nome")
7 ' Corrige os apóstrofos
8 au_nome=Replace(au_nome, "'", "")
9
10 au_ano=request.querystring("ano")
11 au_ID=request.querystring("ID")
12 Set Conn = Server.CreateObject("ADODB.Connection")
13 conn.open "Estudante","Estudante","magic"
14
15 SQLstmt = "UPDATE autores "
16 SQLstmt = SQLstmt & "SET autor=" & au_nome & ", "
17 SQLstmt = SQLstmt & "ano_nasc=" & au_ano
18 SQLstmt = SQLstmt & " WHERE AU_ID=" & au_ID
19 Set RS = Conn.Execute(SQLstmt)
20 If err.number>0 then
21 response.write "Ocorreram erros no script:" & "<P>"
22 response.write "Número=" & err.number & "<P>"
23 response.write "Descrição =" & err.description & "<P>"
24 response.write "Help Context =" & err.helpcontext & "<P>"
25 response.write "Help Path =" & err.helppath & "<P>"
26 response.write "Native Error =" & err.nativeerror & "<P>"
27 response.write "Source =" & err.source & "<P>"
28 response.write "SQLState =" & err.sqlstate & "<P>"
29 else
30 response.write "Nenhum problema aconteceu!" & "<P>"
31 end if
32 IF conn.errors.count> 0 then
33 response.write "Ocorreram erros com o banco de dados" & "<P>"
34 response.write SQLstmt & "<P>"
35 for counter = 0 to conn.errors.count
36 response.write "Número do erro:" & conn.errors(counter).number & "<P>"
37 response.write "Descrição --> " & conn.errors(counter).description & "<P>"
38 next
39 else
40 response.write "Parece que tudo está ok. O Autor foi atualizado!" & "<P>"
41 end if
42 Conn.Close
43 %>
44 </BODY></HTML>
```

## ATUALIZANDO REGISTROS

Esta página demonstra a funcionalidade de se atualizar registros em um banco de dados através de uma consulta SQL. Supomos que o arquivo ASP seja chamado da seguinte forma:

**dbupdate.asp?nome=Alexandre&id=9000&ano=1901**

O script abaixo faz uma atualização no sql através do statement Update. É aconselhável que todas as inclusões, alterações e exclusões no sql seja feita através de Stored Procedures. Assim, fica mais fácil controlar permissões de execução e segurança ainda maior com relação aos códigos.

```
1 <TITLE>dbupdate.asp</TITLE>
2 <body bgcolor="#FFFFFF">
3 <HTML>
4 <%
5 on error resume next
6 auname=request.querystring("nome")
7 auano=request.querystring("ano")
8 auID=request.querystring("ID")
9 Set Conn = Server.CreateObject("ADODB.Connection")
10 conn.open "Estudante","Estudante","magic"
11 SQLstmt = "UPDATE autores "
13 SQLstmt = SQLstmt & "SET autor=" & auname & ","
14 SQLstmt = SQLstmt & "ano_nasc=" & auano
15 SQLstmt = SQLstmt & " WHERE AU_ID=" & auid
16 Set RS = Conn.Execute(SQLstmt)
17 If err.number>0 then
18 response.write "Ocorreram erros no Script:" & "<P>"
19 response.write "Número =" & err.number & "<P>"
20 response.write "Descrição =" & err.description & "<P>"
21 response.write "Help Context =" & err.helpcontext & "<P>"
22 response.write "Help Path =" & err.helppath & "<P>"
23 response.write "Native Error =" & err.nativeerror & "<P>"
24 response.write "Source =" & err.source & "<P>"
25 response.write "SQLState =" & err.sqlstate & "<P>"
26 else
27 response.write "Nenhum problema aconteceu!" & "<P>"
28 end if
29 IF conn.errors.count> 0 then
30 response.write "Ocorreram erros com o banco de dados" & "<P>"
31 for counter = 0 to conn.errors.count
32 response.write "Número " & conn.errors(counter).number & "<P>"
33 response.write "Descrição -> " & conn.errors(counter).description & "<P>"
34 next
35 else
36 response.write "Tudo parece ok. O Autor foi atualizado!" & "<P>"
37 end if
38 Conn.Close
39 %>
40 </BODY></HTML>
```

## COMANDOS AVANÇADOS

### **Funções**

Para simplificar programas

### **Subrotinas**

Para simplificar programas

### **Content Rotator**

Para variar uma informação toda vez que alguém visitar a página.

### **Strings**

Explica algumas funções básicas de manipulação de strings.

### **Rotina de Erro**

Para tratar alguns inconvenientes

### **Dicas para "Debugar"**

Para ajudar na solução de problemas

## FUNÇÕES

O exemplo abaixo é de uma função que recebe um número (mês) e converte para o valor em extenso (nome). A declaração de funções segue o esquema abaixo. Como dissemos anteriormente, é bom ter um arquivo Include com todas as funções usadas por um determinado sistema ou página, semelhante aos módulos do VB.

```
1 <%functions.asp
2 Public Function MesExtenso (mes)
3   Select Case mes
4     Case 1
5       MesExtenso = "Janeiro"
6     Case 2
7       MesExtenso = "Fevereiro"
8     Case 3
9       MesExtenso = "Março"
10    Case 4
11     MesExtenso = "Abril"
12    Case 5
13     MesExtenso = "Maio"
14    Case 6
15     MesExtenso = "Junho"
16    Case 7
17     MesExtenso = "Julho"
18    Case 8
19     MesExtenso = "Agosto"
20    Case 9
21     MesExtenso = "Setembro"
22    Case 10
23     MesExtenso = "Outubro"
24    Case 11
25     MesExtenso = "Novembro"
26    Case 12
27     MesExtenso = "Dezembro"
28  End Select
29 End Function
```

## SUBROTINAS

Subrotinas servem basicamente para poupar a repetição de tarefas. O exemplo abaixo ilustra uma simples subrotina para formatar uma string. Lembrando que as subrotinas também podem ficar nos arquivos tipo Include.

```

1  <%response.buffer=false%>
2  <html><head>
3  <title>subs.asp</title></head>
4  <body bgcolor="#FFFFFF">
5  <br><br>
6  Fazer uma conexão com o banco de dados de contratado, por exemplo.
7  Montar um recordset com o nº do contratado e outras informações quaisquer.
8  Dim NroContratado
9  Nrocontratado = RS("NroContratado")
10 ZerosAEsquerda NroContratado, 8
11 Response.write NroContratado
12
13 ' Acrescenta zeros a esquerda da string até um tamanho máximo para a string
14 '
15 Public Sub ZerosAEsquerda (ByRef strString, intTamString)
16 If IsNull(strString) Then
17   strString = String(intTamString, "0")
18 Else
19   strString = String(intTamString - Len(strString), "0") & strString
20 End If
21 End Sub
22 </body></html>

```

## CONTENT ROTATOR

Esta página demonstra a funcionalidade do componente **Content Rotator**. O script é simples.

```

1  <HTML>
2  <HEAD>
3  <TITLE>cr.asp</TITLE>
4  </HEAD>
5  <body bgcolor="#FFFFFF">
6  <%
7  Set Dica = Server.CreateObject("IISSample.ContentRotator")
8  response.write "Este é o meu conselho:<P>"
9  response.write Dica.ChooseContent("/dicas/dicas.txt")
10 %>
11 </BODY>
12 </HTML>

```

O arquivo Dicas.txt pode se parecer com o seguinte:

```
1 %%#3
2 Cante sempre que estiver tomando banho.
3 %%#3
4 <font size="2" face="ARIAL,HELVETICA">Planeje o seu dia logo no início da
manhã.</font>
5 %%#3
6 <font size="4" face="ARIAL,HELVETICA">Tenha pelo menos 8 horas de sono
por dia!</font>
```

## STRING

As funções para String são muito úteis para passar dados para ASCII, formatar saídas complexas e entradas para formulários, por exemplo. O seguinte script é só uma amostra da função MID. A maioria das funções de formatação de string utilizadas no VB podem ser usadas aqui no ASP. Mais abaixo existem alguns exemplos.

```
1 <title>strings.asp</title>
2 <body bgcolor="#FFFFFF">
3 <% Dim teste
4 teste = "Hello. How are you today Student."5 response.write ("teste =" & teste)
6 response.write ("mid(teste,1,5)=" & mid(teste,1,5))
7 %>
8 </body>
```

### Outras funções do VBScript

Asc, AscB, AscW, Chr, ChrB, ChrW, Filter, Instr, InStrB, InstrRev, Join, Len, LenB, LCase, UCase, Left, LeftB, Mid, MidB, Right, RightB, Replace, Space, Split, StrComp, String, StrReverse, LTrim, RTrim, Trim

## ROTINA DE ERRO

Esta página demonstra uma rotina básica para tentar interpretar alguns erros que podem ser de script ou de banco de dados. Aconselho sempre usar esse tipo de rotina em todos os arquivos ASP que fazem acesso a Banco de Dados. E espero que ele nunca seja executado em seus programas...

```
<% on error resume next
...
Set Conn = Server.CreateObject("ADODB.Connection")
...
SQLstmt = "... "Set RS = Conn.Execute(SQLstmt)
If err.number>0 then%>
  Ocorreram Erros no Script:<P>
  Número do erro =<%=err.number%><P>
  Descrição do erro =<%=err.description%><P>
  Help Context =<%=err.helpcontext%><P>"
  Help Path =<%=err.helppath%><P>
  Native Error =<%=err.nativeerror%><P>
  Source =<%=err.source%><P>
  SQLState =<%=err.sqlstate%><P>
<%else%>
  Nenhum problema aconteceu!<p>
<%end if
IF conn.errors.count> 0 then%>
  Ocorreram erros com o Banco de Dados<P><%=SQLstmt%><P>
  <%for counter = 0 to conn.errors.count%>
    Erro #<%=conn.errors(counter).number%><P>
    Descrição -><%=conn.errors(counter).description%><p>
  <%next
else%>
  Nenhum erro com o Banco de Dados !
<%end if
Conn.Close%>
```

## DICAS PARA "DEBUGAR"

O script abaixo é uma forma geral de debugar problemas.

```
1 <TITLE>dbg.asp</TITLE>
2 <body bgcolor="#FFFFFF">
3 <HTML>
4 <%
5 Response.Write("<P>VARIÁVEIS DO FORMULÁRIO:<br>")
6 Response.Write("-----<br>")
7 For Each Key in Request.Form
8 Response.Write( Key & " = " & Request.Form(Key) & "<br>")
9 Next
10
11 Response.Write("<P>VARIÁVEIS QUERY STRING:<br>")
12 Response.Write("-----<br>")
13 For Each Key in Request.QueryString
14 Response.Write( Key & " = " & Request.QueryString(Key) & "<br>")
15 Next
16
17 Response.Write("<P>VARIÁVEIS TIPO COOKIE:<br>")
18 Response.Write("-----<br>")
19 For Each Key in Request.Cookies
20 Response.Write( Key & " = " & Request.Cookies(Key) & "<br>")
21 Next
22
23 Response.Write("<P>VARIÁVEIS DE SERVIDOR:<br>")
24 Response.Write("-----<br>")
25 For Each Key in Request.ServerVariables
26 Response.Write( Key & " = " & Request.ServerVariables(Key) & "<br>")
27 Next
28 %>
29 </BODY>
30 </HTML>
```

## GERENCIAMENTO DE SESSÕES ASP

Sessões ASP são recursos muito poderosos do ASP. Primeiramente descreveremos o que é e depois como usá-las.

### **O que é uma sessão?**

Provavelmente a primeira questão.

### **Formulário de Usuário e Senha**

Somente se o usuário responder essa questão todas as outras páginas subsequentes poderão ser vistas.

### **Tutorial de Cookie**

Salvar, Mostrar e Deletar cookies.

## O QUE SÃO SESSÕES?

Sessões são uma das facilidades do ASP. Quando alguém visita uma página no seu site, o ASP cria uma sessão e imediatamente pode diferenciar aquele usuário de todos os outros que estão no site. Nada armazenado na sessão daquele usuário pode ser recuperado ou manipulado por uma página e pelas próximas páginas que ele visitar. Algumas coisas devem ser ressaltadas:

Os dados da sessão são armazenados no servidor e não em cookies. Nenhum usuário pode examinar um cookie e determinar o conteúdo de qualquer variável de sessão.

Um cookie é usado para coordenar o session ID do usuário. Novamente, o cookie não contém nenhum dado, somente o session ID. Isto significa que se o usuário não aceita cookies, você não poderá usar as sessões como descrito nesse tutorial.

Se você realmente precisa usar sessões sem o browser cliente permitir os cookies, instale um filtro ISAPI chamado "Cookie Mungger" que resolverá o seu problema, mas a performance cairá bastante. Este filtro está disponível no endereço:

<http://backoffice.microsoft.com/downtrial/moreinfo/iissamples.asp>

## FORMULÁRIO USUÁRIO E SENHA

Os formulários abaixo estão usando o método GET no envio das informações. Como vimos anteriormente para acessar uma informação enviada por formulário com método GET, usamos: `request.querystring("nome_do_campo")`.

Quando usamos o método POST, usamos: `request.form("nome_do_campo")`

### Exemplo 1: Formulário padrão de Usuário e Senha

```
1 <html>
2 <head>
3 <title>sessiontest01.asp</title>
4 </head>
5 <body bgcolor="#FFFFFF">
6 </body>
7 </html>
8 <html>
9
10 <body>
11 <form name="meu_autor" action="sessiontest02.asp"
12 method="GET">
13 <p>Seu Nome: <input type="TEXT" name="seu_nome"></p>
14 <p> Sua Senha: <input type="TEXT" name="sua_senha"></p>
15 <p> <input type="submit"> </p>
16 </form>
17 </body>
```

**Exemplo 2: Armazena um Usuário e Senha na sessão**

```
1 <html>
2 <head>
3 <title>sessiontest02.asp</title>
4 </head>
5 <body bgcolor="#FFFFFF">
6 <% ' Programa ASP que armazena um usuário e uma senha na sessão
7 senha_usuario=request.querystring("sua_senha")
8 nome_usuario=request.querystring("seu_nome")
9 response.write "Fala " & nome_usuario & "<P>"
10 select case senha_usuario
11 case "666"
12 response.write "Sua senha é o número da besta"
13 case else
14 response.write "Senha não reconhecida!"
15 end select
16 session("who")=nome_usuario
17 session("securitypassword")=senha_usuario
18 %>
19 </body>
20 </html>
```

**Exemplo 3: Lê as variáveis da sessão**

```
1 <html>
2 <head>
3 <title>sessiontest03.asp</title>
4 </head>
5 <body bgcolor="#FFFFFF">
6 <%
7 ' Programa ASP que lê variáveis da sessão
8 response.write "Fala " & session("who") & ".<P>Obrigado por passar aqui.<P>"
9 If session("securitypassword")="666" then
10 response.write "Encontro secreto na sala do tio Bill"
11 else
12 response.write "A sala do tio Bill está em reformas. Os encontros foram
cancelados."
13 end if
14 %>
15 </body>
16 </html>
```

## ARMAZENANDO COOKIES

Formulário que recebe cookies do usuário e repassa para armazenamento.

```
1  <%
2  response.buffer=true
3  %>
4  <html><head>
5  <TITLE>cookiesform.asp</TITLE></head>
6  <body bgcolor="#FFFFFF">
7  <%
8  un=Request.Cookies("Pessoa")("ult_nome")
9  pn=Request.Cookies("Pessoa")("primeiro_nome")
10 es=Request.Cookies("Pessoa")("estado")
11 %>
12 <Form action = "cookiesformrespond.asp">
13 Formulário com Cookies<p>
14 Por favor, entre o seu primeiro nome<p>
15 <Input NAME="primeironome" size = "40" value=<%=pn%>>
16 <p>
17 Por favor entre o seu último nome<p>
18 <Input NAME="ultimonome" size = "40" value=<%=un%>>
19 <p>
20 Por favor entre a abreviação do seu Estado<p>
21 <Input NAME="estado" MaxLength="2" value=<%=es%>>
22 <Input type=submit></form>
23 </body></html>
```