Melhorando a performance de suas páginas ASP

José Francisco do Santos Neto *

Introdução

Para velhos programadores como eu, o primeiro contato com o desenvolvimento de aplicações para a Internet é, no mínimo, curioso. A primeira visão que temos é obviamente simplista e precipitada, mas se resume em: "Estamos desenvolvendo uma aplicação muito simples, com uma linguagem muito simples, mas que poderá funcionar em qualquer máquina do mundo."

A última afirmação é o que nos leva a seguir em frente, não somente para constatarmos o quão errada é esta visão precipitada, mas também para nos apaixonarmos cada vez mais pela gama de possibilidades que o desenvolvimento para a Internet nos traz.

A tecnologia Active Server Pages (ASP) tem sido amplamente usada em sites importantes por todo o mundo. ASP fornece o ambiente necessário para:

- executar scripts no servidor;
- usar componentes do servidor e
- acessar banco de dados no servidor.

Dessa maneira, usando ASP podemos criar aplicações com tecnologia cliente-servidor suportando múltiplos acessos simultâneos através de uma linguagem de programação muito simples.

Nesta matéria, discutiremos uma série de tópicos relacionados com a otimização da performance de aplicações construídas em ASP.

É importante lembrar que qualquer ganho em performance no ambiente da Internet é extremamente importante, pois estamos falando de uma aplicação que poderá estar funcionando com dezenas, centenas ou até milhares de acessos, em um espaço de tempo muito curto. Sem falar do fato de que esses acessos utilizam uma conexão remota (que em muitos casos já é lenta por natureza!!!).

As técnicas abordadas nesta matéria já foram amplamente testadas por diversos especialistas em ASP. Portanto, não me preocuparei em falar das opções não indicadas, e sim nas melhores formas para se construir páginas ASP.

Divisão por Assuntos

Muitos são os tópicos de que iremos tratar nesta matéria. Dessa forma, resolvi agrupá-los em três assuntos:

- Páginas ASP;
- Uso de Componentes e
- Acesso a Dados.

Páginas ASP

Gerando Conteúdo com ASP

Uma das justificativas mais importantes para o uso de ASP é a geração de conteúdo dinâmico. Existem algumas formas para fazê-lo, e a mais indicada é usar o método Write do objeto Response (Response.Write) em uma única instrução como mostra o seguinte exemplo:

```
...
Response.Write("<html>" & _
  "<head>" & _
  "<title>Gerando Conteúdo com ASP</title>" & _
  "</head>" & _
  "<body>" & _
  "<h1>Gerand Conteúdo com ASP</h1>" & _
  "<table>" & _
```

```
  "<tr><td><b>Nome:</b></td><td>" & strNome & "</td></tr>" & _
  "<tr><td><b>Aniverssário:</b></td><td>" & dtmAniverssario & "</td></tr>" & _
  "</table>" & _
  "</body>" &  _
  "</html>")
...
```

Isto se justifica pelo fato de que o interpretador ASP precisa apenas interpretar uma única instrução para exibir o conteúdo desejado.

Uso de Quebras de Linha

Quando se envia conteúdo para o browser da maneira descrita anteriormente, todo o conteúdo fica em uma única linha no código fonte exibido no browser. Se você se preocupa em como o código será exibido no browser, você poderá utilizar a constante vbCrLf concatenada ao conteúdo enviado através do método Write para inserir quebras de linha. Isto acarretará uma pequena queda na performance, mas que é suportável nos casos onde a formatação do código fonte é importante, como mostra o seguinte trecho de código:

```
...
Response.Write("<html>" & vbCrLf & _
  "<head>" & vbCrLf & _
  "   <title>Gerando Conteúdo com ASP</title>" & vbCrLf & _
  "</head>" & vbCrLf & _
...
```

Uso de Comentários no Código

O uso de comentários em páginas HTML não é algo muito interessante pois aumenta a quantidade de informações que é transferida do Servidor Web para o Browser, e só tem utilidade para os programadores da página. Entretanto, os comentários no código ASP não representam uma mudança significativa na performance das páginas pois são simplesmente ignorados pelo interpretador e não são transferidos para o browser.

Scripts Muito Longos

Obviamente, você pode incluir suas regras de negócio em páginas ASP, pois elas são processadas no Servidor Web. Entretanto, isto pode tornar os seus scripts muito longos.

Não podemos perder de vista que todo código ASP é interpretado linha a linha. Dessa maneira, scripts muito longos podem se tornar muito lentos. Nestes casos, a melhor maneira é usar Componentes que encapsulem as regras de negócio em código compilado.

Arquivos "#include" Muito Longos

Os arquivos de include são uma ótima opção para economizar linhas de código que podem ser reutilizadas em diversas páginas. Entretanto, existe uma pequena diminuição na performance de páginas ASP que usam includes. Principalmente se eles forem muito longos. Mesmo que você esteja usando apenas parte de um arquivo de include, como nos casos onde eles funcionam como uma biblioteca de rotinas, todo o arquivo de include é interpretado antes da informação chegar até o browser.

Este é mais um motivo que nos motiva a usar componentes compilados em páginas ASP.

Uso de Variáveis Globais

Sempre que possível, evite o uso de variáveis globais. As variáveis declaradas dentro de subrotinas ou funções são acessadas mais rapidamente e deixam o código mais limpo e organizado.

Uso de Option Explicit

Além de ser uma ótima prática de programação, o uso da instrução Option Explicit, que obriga a declaração de variáveis, torna o processamento da página mais rápido.

Uso de Blocos de Função

O uso de Blocos de Função diminui a performance das páginas ASP devido às chamadas extras que se fazem necessárias a esses blocos. Entretanto, o fato de nesses casos algumas variáveis se tornarem locais dos blocos, faz com que seja vantajoso o uso de Blocos de Função.

Dessa maneira, sempre que se fizer necessário o uso de blocos de função para economizar código, faça-o e use variáveis locais sempre que possível.

Tratamento de Erro

Tratamento de Erros é essencial em aplicações sérias. Mas existe uma pequena queda na performance das páginas que possuem tratamento de erro. Dessa maneira, o bom senso diz que o tratamento de erro deve ser usado sempre que você não tiver controle sobre os erros que possam acontecer em sua aplicação. Um bom exemplo é quando você usa objetos que acessam outros recursos, tais como ADO ou FileSystem.

Uso de Transações

O uso de transações (<%@ TRANSACTION = REQUIRED %>) causa uma queda dramática da performance de páginas ASP. Dessa forma, apenas use transações quando duas ou mais operações precisem ser realizadas de uma única vez.

Uso de código ASP e HTML Alternado

Evite intercalar código ASP com código HTML, como no exemplo que se segue:

```
...
<% For intContador = 1 To 10 %>
<TR><TD>Contando ... <%= intContador %></TD></TR>
<% Next %>
...
```

Preferencialmente, agrupe o código ASP em blocos sempre que possível:

```
...
<%
  For intContador = 1 To 10
     Response.Write "<TR><TD>Contando ... " & intContador & "</TD></TR>"
  Next
%>
...
```

Uso de Buffer

Você pode determinar que o conteúdo de sua página ASP seja exibido somente depois de todo o processamento da página através da propriedade Response.Buffer = True.

Isto é uma boa prática, pois aumenta a performance de processamento das páginas ASP. Mas em alguns casos, é necessário enviar alguma informação ao cliente durante o processamento das páginas. Nestes casos, use Response.Buffer = False ou use o método Response.Flush para enviar as informações processadas até o momento para o browser.

Desabilitar o Controle de Estado Quando Possível (Local e Server)

O controle de estado é algo muito interessante que as páginas ASP nos proporcionam. Entretanto, existe uma queda na performance de páginas que fazem uso deste recurso.

Desta maneira, sempre que o controle de estado não se fizer necessário em uma página, use a seguinte instrução para desabilitá-lo: <%@ENABLESESSIONSTATE = FALSE %>.

Se nenhuma das páginas de seu site fizer uso de controle de estado, você pode configurar diretamente nas propriedades do site. Para isto, selecione a caixa de diálogo *Properties* do site em questão. Em seguida, selecione o botão *Configuration* na aba *Home Directory*. Desmarque, então, a opção *enable session state em App options*.

Uso de Arrays Dinâmicos

O uso da instrução Redim nos economiza um pouco de memória e a preocupação de estourar o número limite de elementos dos arrays em nossos códigos. Entretanto, o uso de Redim provoca uma queda na performance. O ideal é usar a instrução Dim já com a quantidade de elementos que será usada. Mesmo nos casos em que você não sabe a quantidade exata, o uso de uma ordem de grandeza, com o risco de ocupar alguns espaços desnecessários da memória, compensa em função da velocidade.

Uso da Propriedade IsClientConnected

Sempre que sua página ASP tiver um longo processamento, use o método **Response.IsClientConnected** para verificar se o cliente ainda está conectado. Caso contrário, você poderá abortar o processamento evitando, assim, o consumo desnecessário de recursos do servidor.

Uso do Mesmo Objeto Muitas Vezes

Se você for usar o mesmo objeto muitas vezes e estiver usando o VBScript 5.0, faça uso da instrução **With** para que não seja necessário requalificar o objeto a cada propriedade ou método usado:

```
...
With Response
  .Write "Linha 1<BR>"
  .Write "Linha 2<BR>"
  .Write "Linha 3<BR>"
  .Flush
  .Write "Linha 4<BR>"
  .Write "Linha 5<BR>"
  .Write "Linha 6<BR>"
End With
...
```

Eventos Session_OnStart ou Session_OnEnd Sem Conteúdo

Remova eventos do objeto Session que não tenham conteúdo. Isto economizará passos do interpretador, aumentando a performance.

Uso do Visual Basic

Se você estiver usando o Visual Basic para construir seus componentes, aqui vão algumas digas para aumentar a performance dos mesmos:

- Use o Visual Basic 6.0 com o Service Pack 3.
- Use as opções *Unattended Execution* e *Retain in Memory* em seus projetos (figura 1) e evite o uso de componentes criados com o VB em ambientes *Multi-threaded*.

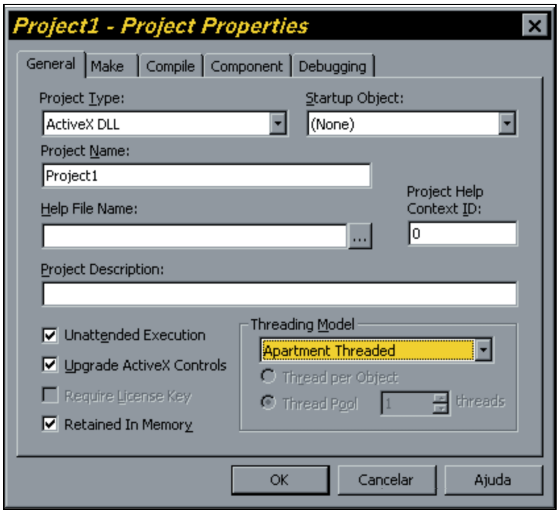

Fig. 1 - Use as opções Unattended Execution e Retain in Memory

Acesso aos Dados

Uso de Índices

Sempre que possível, faça uso de índices em seus bancos de dados. Os índices aumentam incrivelmente a performance de consultas e stored procedures acessadas via ADO em páginas ASP. O uso de índices é recomendado principalmente se você estiver usando o SQL Server.

Retornando Apenas o Necessário

Certifique-se que suas consultas estão retornando apenas os campos necessários. Nos casos de consultas que retornam muitos registros, tente paginar a consulta de forma que ela retorne apenas alguns registros de cada vez. Isto poderá aumentar bastante a performance de suas páginas ASP.

Versão do Microsoft Data Access Components (MDAC)

Procure usar o MDAC 2.1 com Service Pack 2 ou superior. O Ideal é usar o MDAC 2.5.

Armazenando Connections ADO em Variáveis de Aplicação ou de Sessão

Jamais armazene objetos Connection em variáveis de Aplicação ou de Sessão. O ADO já possui um mecanismo chamado *pooling* que otimiza o uso de instâncias de conexões. Além do mais, como foi visto anteriormente, não é uma boa idéia armazenar instâncias de objetos em variáveis de Sessão.

Instancie suas Conexões diretamente na página em questão e não esqueça de atribuir Nothing a elas quando terminar o uso.

Fechar Explicitamente Recordsets e Connections

Tanto objetos Recordset como Connection devem ser fechados explicitamente através do método *Close* assim que termine o seu uso. Desta forma, você os disponibiliza para o pooling do ADO.

Reutilização de Recordsets e Commands

Evite a reutilização de objetos Recordset e Command. Isto não necessariamente irá aumentar a performance de suas aplicações, mas sua aplicação ficará mais clara e mais fácil de dar manutenção.

Utilizando o Tipo de Cursor (Cursor Type) e o Tipo de Bloqueio (Lock Type) Mais Eficientes

Use sempre o tipo de cursor e o tipo de bloqueio mais simples para a tarefa que você deseja realizar.

A tabela 1 mostra em ordem decrescente de performance os tipos de cursor e tipos de bloqueio.

Cursor Types	Lock Types
Forwardonly	Batchoptimistic
Static	Optimistic

Dynamic	Pessimistic
Keyset	

Tabela 1 - Tipos de cursor e tipos de bloqueio em ordem decrescente de performance

Uso de GetRows

O método GetRows atribui todo o conteúdo de um Recordset para um vetor bidimensional, cuja primeira dimensão representa os campos e a segunda representa os registros. Logo em seguida, o Recordset pode ser fechado. O uso de GetRows aumenta muito a performance de páginas ASP:

```
...
If objRS.EOF Then
  Response.Write ("Não foram encontrados registros.")
  objRS.Close
  Set objRS = Nothing
Else
  Dim arrRS
  arrRS = objRS.GetRows
  objRS.Close
  Set objRS = Nothing

  Dim NumRegistros
  Dim NumCampos
  Dim RegistroAtual
  Dim CampoAtual

NumCampos = Ubound(arrRS, 1)
NumRegistros = Ubound(arrRS, 2)
Response.Write "<TABLE>"
For RegistroAtual = 0 To NumRegistros
    Response.Write "<TR>"
    For CampoAtual = 0 To NumCampos
      Response.Write "<TD>" & arrRS(CampoAtual, RegistroAtual) & "</TD>"
    Next
    Response.Write "</TR>"
  Next
  Response.Write "</TABLE>"
End If
...
```

Conexões Sem DSN

As conexões sem DSN normalmente são mais rápidas que as conexões usando DSN de sistema, que por sua vez são mais rápidas que as conexões usando DSN de arquivo.

As conexões em DSN são aquelas cujos parâmetros de conexão são passados diretamente no momento da criação da conexão.

Uso do Access

Se você estiver usando o Access como base de dados, não espere grandes ganhos de performance já que ele é um banco de dados baseado em arquivo e não é cliente servidor.

Uso do Provider OLEDB SQL (SQLOLEDB)

Se você estiver usando o SQL Server, use o provider SQLOLEDB para aumentar a performance de sua aplicação.

Uso do arquivo ADOVBS.inc

Evite o uso do arquivo ADOVBS.inc que contém as declarações das constantes usadas pelo ADO. Ao invés disso, declare apenas as constantes que você irá usar ou coloque a seguinte referência em seu arquivo global.asa:

```
<!--METADATA TYPE="typelib" FILE="c:\arquivos de programas\arquivos comuns\system\ado\msado15.dll" NAME="ADODB Type
Library" -->
```

ou

```
<!--METADATA TYPE="typelib" UUID="00000205-0000-0010-8000-00AA006D2EA4" NAME="ADODB Type Library" -->
```

ou uma versão superior.

Uso de um Objeto Connection Separado Quando Trabalhar com Recordsets

Se você estiver usando um único Recordset, passe a string de conexão diretamente para a propriedade *ActiveConnection*:

```
objRS.ActiveConnection = Application("ConexaoString")
```

Quando você estiver utilizando vários Recordsets, crie um objeto Connection e use-o na propriedade *Active Connection*:

```
...
Set objConn = Server.CreateObject("ADODB.Connection")
objConn.Open Application("ConexaoString")

Dim intContador
For intContador = 1 To 10

Set objRS = Server.CreateObject("ADODB.Recordset")
Set objRS.ActiveConnection = objConnection
objRS.CursorType = 0 'adOpenForwardOnly
objRS.LockType = 1 'adLockReadOnly
objRS.Open Application("SQL")

If objRS.EOF Then
  Response.Write "Não Foraam Encontrados Registros"
Else
    'Escreve o Conteúdo
    ...
  End If

objRS.Close
Set objRS = Nothing
Next
objConn.Close
Set objConn = Nothing
...
```

Conclusão

O ganho de performance de cada um dos itens abordados pode não ser significante quando vistos isoladamente. Mas o uso conjunto dessas dicas pode trazer um aumento expressivo na performance de suas aplicações ASP.

Em uma próxima matéria, veremos como medir a performance de páginas ASP usando uma ferramenta fornecida gratuitamente pela Microsoft que funciona com o IIS.

Abraços e até a próxima.

* **José Francisco do Santos Neto** é colaborador do Boletim Desenvolvedor Web