Cetec
Capacitações

Centro
Paula Souza

GOVERNO DO ESTADO
SÃO PAULO

GOVERNO FEDERAL
BRASIL
PÁTRIA EDUCADORA

- Sessões
  - ✓ Global.asax
- Cookies
  - ✓ Gravar
  - ✓ Ler
  - ✓ Excluir

Session

Toda vez que um usuário Web se conecta a um aplicativo ASP é iniciada uma sessão para o mesmo no servidor Web. Para representar tal sessão, o ASP possui um objeto interno chamado Session. Esse objeto pode armazenar valores ligados apenas a um único visitante do site(o dono da sessão). Com ele podemos criar variáveis de qualquer subtipo cujo valor pode ser acessado ou modificado somente pelo "dono" da sessão.

Para criar uma variável do nível de sessão, devemos escrever comandos com seguinte sintaxe:

Session["NOME_DA_VARIAVEL"] = VALOR_DA_VARIAVEL

As variáveis de sessão permanecerão na memória (ativas) até a sessão ser encerrada. Isso pode acontecer quando o usuário fechar o web browser, quando ocorre o "TIMEOUT" da sessão, ou quando o script invoca o método ABANDON do objeto Session. A propriedade TIMEOUT é usada quando o usuário fica parado sem fazer nada no Browser.

O default é vinte minutos, mas esse valor pode ser modificado da seguinte forma:

Session.Timeout = VALOR_MINUTOS

Ainda relacionado a esse objeto existem dois eventos:

Session_OnStart ➔ Ocorre quando a sessão é iniciada

Session_OnEnd ➔ Ocorre quando a sessão é finalizada.

Global.asax

Cada aplicativo construído na arquitetura ASP.NET, pode conter um arquivo chamado **Global.asax**. O Global.asax é uma arquivo de texto com as mesmas características de um arquivo .aspx, ele é responsável pela configuração de eventos de aplicação e de sessão. Ele é um arquivo opcional, o que significa que, podemos criar aplicações onde os eventos de aplicação e de sessão não serão tratados.

O Global.asax, possui uma estrutura e uma sintaxe bastante semelhante ao antigo Global.asa. Como o ASP.NET consegue "rodar" código ASP antigo, para manter a compatibilidade com sistemas existentes, da mesma forma podemos ter os arquivos Global.asax e Global.asa, em uma mesma aplicação. Mas nessas situações - os dois arquivos coexistindo -, quando uma página .aspx for solicitada pelo usuário, é o arquivo Global.asax que será processado, e vice-versa.

Como veremos mais adiante, o Global.asax pode conter códigos, que serão executados em respostas a eventos (de aplicação ou de sessão). Como qualquer outro arquivo de código, em .NET, o Global.asax será compilado para IL. A compilação dos arquivos de configuração ocorre sempre que a aplicação é iniciada, e só será recompilado quando houver modificações.

O arquivo Global.asax, deve ser criado no diretório raiz da aplicação. Se o arquivo for gerado em outro sub diretório ele será ignorado.

Estrutura de um Global.asax

A estrutura de um arquivo Global.asax, é composta de elementos que assim como o próprio arquivo, são opcionais. Sendo assim, os elementos omitidos na construção do arquivo, apenas deixaram de ser tratados. A construção do Global.asax é composta pelos seguintes elementos:

- ***Diretivas de Aplicação*:** Essas diretivas de aplicação, modificam ou acrescentam parâmetros a compilação de páginas ASP.NET.
- ***Implementação de código*:** Podemos definir métodos que responderam aos eventos gerados pela aplicação.
- ***Declaração de Objetos*:** Neste elemento, temos a possibilidade de declarar variáveis de sessão em um formato de tag, semelhante a declaração de WebControls em páginas ASP.NET.
- ***Server-Side Include*:** É possível especificar arquivos à serem incluídos dentro do Global.asax.

Diretivas de Aplicação

As diretivas de aplicação são semelhantes às encontradas em páginas .aspx. As diretivas devem ser declaradas no início dos arquivos Global.asax. Observe o código abaixo a utilização da diretiva **Import**, que importa o Namespace "System.Data.OleDb".

```
<%@ Import Namespace="System.Data.OleDb" %>
```

Quando importamos um Namespace dentro do Global.asax, estamos disponibilizando as classes do Namespace para todas as páginas da aplicação, automatizando a tarefa de importar o mesmo Namespace em todas as páginas. Na declaração abaixo estaremos definindo a diretiva **Application**, com duas opções **Inherits** e **Description**. O atributo **Inherits** nos permite definir uma classe, da qual o Global.asax irá herdar suas funcionalidades (como padrão o Global.asax deriva da classe System.Web.HttpApplication).

```
<%@ Application Inherits="Namespace.Classe" %>
```

O outro atributo da diretiva **Application**, serve para definir uma descrição da aplicação. O **Description** pode ser utilizado como um simples comentário:

```
<%@ Application Description="Exemplo - Porta WebMobile" %>
```

A diretiva **Assembly** serve para importar uma DLL, disponibilizando o seu código dentro do Global.asax. Isso pode ser necessário quando trabalhamos com a arquitetura de três camadas, em que temos código encapsulado em um Assembly.

```
<%@ Assembly Name="Componente.dll" %>
```

Implementação de código

Este elemento do Global.asax, nos permite escrever código em resposta aos eventos gerados pela aplicação. A estrutura de um bloco de código no Global.asax é semelhante aos códigos implementados em páginas .aspx, e devem seguir o formato a seguir:

```
<script Language="Linguagem" src="Arquivo externo" runat="server">
    ...
    Código
    ...
</script>
```

Na arquitetura do ASP.NET, novos eventos foram criados, nos dando a possibilidade de trata-los dentro do Global.asax. No ASP antigo, estes eventos eram tratados nas próprias páginas.

Segue abaixo uma listagem de eventos que podem ser tratados no ASP.NET.

```
<script Language="C#" runAt="server">
    protected void Application_Start(Object sender, EventArgs e)
    {
       //Evento disparado quando a aplicação é iniciada.
    }

    protected void Session_Start(Object sender, EventArgs e)
    {
       //Evento disparado quando a uma sessão é iniciada.
    }

    protected void Application_BeginRequest(Object sender, EventArgs e)
    {
       //Evento disparado antes da requisição de uma página.
    }
```

```
protected void Application_EndRequest(Object sender, EventArgs e)
{
   //Evento disparado depois da requisição de uma página.
}

protected void Application_AuthenticateRequest(Object sender, EventArgs e)
{
   //Evento disparado quando uma requisição feira por um usuário é
   autorizada.
}

protected void Application_Error(Object sender, EventArgs e)
{
   //Evento disparado quando uma ocorre uma exceção na aplicação.
}
```

```
    protected void Session_End(Object sender, EventArgs e)
    {
       //Evento disparado quando a sessão é finalizada.
    }

    protected void Application_End(Object sender, EventArgs e)
    {
       //Evento disparado quando uma aplicação é finalizada.
    }
</script>
```

Declaração de Objetos

Dentro do Global.asax, temos a possibilidade de declarar objetos, que pertencerão à aplicação, ou à sessão do usuário. Estes objetos deveram ser declarados no formato de tag - <object> </object> -. Observe o seguinte bloco de código, vamos declarar uma objeto DataSet que pertencerá a sessão do usuário:

```
<object id="objDS" class="System.Data.DataSet" scope="Session" runat="server">
</object>
```

Como citado acima, o objeto objDS será disponibilizado na sessão do usuário. Dessa forma, poderemos acessa-lo pelo objto Session como no código a seguir:

```
DataSet objDS = Session["objDS"] as DataSet;
```

Server-Side Include

Este elemento do **Global.asax,** permite a inclusão de códigos de arquivos externos em páginas .aspx. A utilização do Include no ASP.NET, é semelhante à no ASP antigo. Um arquivo pode ser incluído de duas maneiras, pelo seu path (caminho) ou URL. Os arquivos do tipo Include devem ter a extensão **.inc** ou **.aspx.** Observe a utilização deste recurso no código abaixo:

```
<!--#include file="teste.inc" --> ou <!--#include virtual="teste.aspx" -->
```

Os arquivos do tipo Include são úteis, quando queremos definir um template a ser utilizado em outras páginas ou projetos. Os arquivos do tipo Include são processados antes de serem dispostos nas páginas aspx.

Cookies

Um cookie é um pequeno pedaço de texto que acompanha solicitações e páginas quando elas andam entre o servidor Web e o browser. O cookie contêm informações que o aplicativo Web pode ler sempre que o usuário visitar o site.

Por exemplo, se um usuário solicita uma página do seu site e seu aplicativo envia não apenas uma página, mas também um cookie contendo a data e a hora, quando o navegador do usuário receber a página, ele também recebe o cookie, que ele armazena em uma pasta no disco rígido do usuário.

Mais tarde, se o usuário solicita uma página do seu site novamente, quando o usuário digita a URL, o navegador procura no disco rígido local para um cookie associado ao URL. Se o cookie existe, o navegador envia o cookie para o seu site, juntamente com a solicitação de página. Seu aplicativo pode então determinar a data e hora que o usuário visitou pela última vez o site. Você pode usar as informações para exibir uma mensagem para o usuário ou verificar uma data de validade.

Cookies estão associados com um site Web, não com uma página específica, portanto, o navegador e o servidor vão trocar informações de cookie independentemente de qual for a página que o usuário solicite de seu site. Como o usuário visita diferentes sites, cada site pode enviar um cookie para o navegador do usuário igualmente; o navegador armazena todos os cookies separadamente.

Cookies

Cookies ajudam sites Web a armazenar informações sobre os visitantes. De forma mais geral, cookies são um meio de manter continuidade em um aplicativo Web—isto é, de realizar gerenciamento de estado. Exceto pelo curto tempo enquanto eles estão realmente trocando informação, o navegador e o servidor Web estão desconectados. Cada solicitação que um usuário faz a um servidor Web é tratada independentemente de qualquer outra solicitação. Muitas vezes, entretanto, é útil para o servidor Web identificar usuários quando eles solicitam uma página. Por exemplo, o servidor Web de um site de compras mantém o caminho de compradores individuais, de forma que o site pode gerenciar os carrinhos de compra e outra informação de um usuário específico. Um cookie então atua como um tipo de cartão de chamada, apresentando identificação pertinente que ajuda um aplicativo a saber como proceder.

Cookies são usados para muitos propósitos, todos relativos a ajudar o site Web a lembrar os usuários. Por exemplo, um site realizando uma pesquisa pode usar um cookie simplesmente como um valor Booleano para indicar se um navegador do usuário já participou na votação, a fim de que o usuário não possa votar duas vezes. Um site que pede a um usuário para fazer logon pode usar um cookie para registrar que o usuário já logou nele, de modo que o usuário não tenha que inserir suas credenciais.

Limitações de Cookie

A maioria dos navegadores suportam cookies de até 4096 bytes. Devido a esse limite pequeno, de cookies são melhor usados para armazenar pequenas quantidades de dados ou melhor ainda, um identificador, sistema autônomo uma ID de usuário. A ID de usuário, em seguida, pode ser usada para identificar o usuário e ler as informações de usuário de um banco de dados ou Outros armazenamento de dados. (Veja a seção "Cookies e Segurança" abaixo para informação sobre implicações de segurança de armazenamento de informação de usuário.)

Navegadores também impõem limitações em quantos cookies seu site pode armazenar no computador do usuário. A maioria dos navegadores permite somente 20 cookies por site; se você tentar armazenar mais, os cookies mais antigos são descartados. Alguns navegadores também põem um limite absoluto, geralmente 300, no número de cookies que serão aceitos de todos os sites combinados.

Uma limitação de cookie que você pode encontrar é que os usuários podem configurar seu navegador para recusar cookies. Se você definir um política de privacidade P3P e localizá-la na raiz de seu site Web, muitos navegadores irão aceitar cookies de seu site. Entretanto, você pode ter que evitar os cookies completamente e usar um mecanismo diferente para armazenar informações específicas de usuário. Um método comum para armazenar informações de usuário é estado de sessão, mas estado de sessão depende de cookies, como explicado antes na seção "Cookies e Estado de Sessão".

Gravando Cookie

O navegador é responsável por gerenciar cookies em um sistema do usuário. Cookies são enviados ao navegador através do objeto **HttpResponse** que expõe um coleção chamada Cookies. Você pode acessar o objeto **HttpResponse** como a propriedade **Response** de sua classe **Page**. Quaisquer cookies que você desejar enviar ao navegador deve ser adicionado a esta coleção. Quando criar um cookie, você especifica um **Name** e **Value**. Cada cookie deve ter um nome exclusivo, de forma que ele possa ser identificado posteriormente quando lido pelo navegador. Devido aos cookies serem armazenados pelo nome, nomear dois cookies de forma igual irá fazer com que um seja sobrescrito.

Você pode também definir uma data e hora de validade do cookie. Cookies expirados são excluídos pelo navegador quando um usuário visita o site que gravou os cookies. A validade de um cookie deve ser definida durante o tempo que seu aplicativo considera que o valor do cookie seja válido. Para um cookie nunca expirar efetivamente, você pode definir a data de validade para ser 50 anos a partir de agora.

Se você não definir a validade do cookie, o cookie é criado mas ele não é armazenado no disco rígido do usuário. Em vez disso, o cookie é mantido como parte das informações de sessão do usuário. Quando o usuário fecha o navegador, o cookie é descartado. Um cookie não persistente como este é útil para informação que precisa ser armazenada por um curto período de tempo apenas ou que, por razões de segurança, não deve ser gravado no disco do computador cliente.

Gravando Cookie

Por exemplo, cookies não persistentes são úteis se o usuário estiver trabalhando em um computador público, onde você não deseja gravar o cookie no disco.

Você pode adicionar cookies à coleção **Cookies** de diversas maneiras. O exemplo a seguir mostra dois métodos para gravar cookies:

```
Response.Cookies[“usrNome"].Value = “Capacitacao";
Response.Cookies["usrNome"].Expires = DateTime.Now.AddDays(1);

HttpCookie aCookie = new HttpCookie(“ultimaVisita");
aCookie.Value = DateTime.Now.ToString();
aCookie.Expires = DateTime.Now.AddDays(1);
Response.Cookies.Add(aCookie);
```

O exemplo adiciona dois cookies para a coleção **Cookies**, um chamado **usrName** e o outro chamado **ultimaVisita**. Para o primeiro cookie, os valores da coleção **Cookies** são definidos diretamente. Você pode adicionar valores para a coleção dessa maneira porque **Cookies** deriva de uma coleção especializada do tipo **NameObjectCollectionBase**.

Gravando Cookie

Para o segundo cookie, o código cria uma instância de um objeto do tipo **HttpCookie**, define suas propriedades e, em seguida, adiciona-o à coleção **Cookies** via o método **Add**. Quando você instanciar um objeto **HttpCookie**, você deve passar o nome do cookie como parte do construtor.

Dois exemplos realizam a mesma tarefa, gravar um cookie para o navegador. Em ambos os métodos, o valor de validade deve ser do tipo **DateTime**. No entanto, o valor **ultimaVisita** também é um valor de data - tempo. Devido a todos os valores de cookie serem armazenados como sequências de caracteres, o valor a data - tempo deve ser convertido em um **String**.

Cookies com mais de um valor

Você pode armazenar um valor em um cookie, tal como nome de usuário e última visita. Você também pode armazenar vários pares nome-valor em um cookie único. Os pares nome-valor são denominados subchaves. (Subchaves são enunciados bem como uma sequência de caracteres da consulta em uma URL.) Por exemplo, em vez de criar dois cookies separados chamados usrNome e ultimaVisita, você pode criar um único cookie chamado usrInfo que tenha as subchaves usrNome e ultimaVisita.

Cookies com mais de um valor

Você pode usar subchaves por vários motivos. Primeiro, é conveniente colocar informações relacionadas ou semelhantes em um único cookie. Além disso, devido a todas as informações estarem em um único cookie, atributos de cookie, tais como validade, se aplicam a todas as informações. (Por outro lado, se você desejar atribuir diferentes datas de validade para diferentes tipos de informação, você deve armazenar as informações em cookies separados.)

Um cookie com subchaves também ajuda a limitar o tamanho dos arquivos de cookie. Os cookies são normalmente limitados a 4096 bytes e você não pode armazenar mais que 20 cookies por site. Usando um único cookie com subchaves, você usa menos que esses 20 cookies a que seu site está destinado. Além disso, um único cookie ocupa cerca de 50 caracteres para a sobrecarga (informações de validade e assim por diante) mais o comprimento do valor que você armazena nele, que contam para o limite de 4096 bytes. Se você armazenar cinco subchaves em vez de cinco cookies separados, você salva a sobrecarga dos cookies separados e pode economizar aproximadamente 200 bytes.

Cookies com mais de um valor

Para criar um cookie com subchaves, você pode usar uma variação da sintaxe para gravar um único cookie. O exemplo a seguir mostra duas maneiras para gravar o mesmo cookie, cada um com duas subchaves:

```
Response.Cookies["usrInfo"]["usrNome"] = “Capacitacao";
Response.Cookies["usrInfo"][“ultimaVisita"] = DateTime.Now.ToString();
Response.Cookies["usrInfo"].Expires = DateTime.Now.AddDays(1);

HttpCookie aCookie = new HttpCookie("usrInfo");
aCookie.Values["userName"] = “Capacitacao";
aCookie.Values[“ultimaVisita"] = DateTime.Now.ToString();
aCookie.Expires = DateTime.Now.AddDays(1);
Response.Cookies.Add(aCookie);
```

Lendo Cookies

Quando um navegador faz uma solicitação para o servidor, ele envia os cookies para o servidor junto com a solicitação. Em seus aplicativos ASP.NET, você pode ler os cookies usando o objeto **HttpRequest**, que está disponível como a propriedade **Request** de sua classe **Page**. A estrutura do objeto **HttpRequest** é essencialmente a mesma do objeto **HttpResponse**, portanto, você pode ler cookies fora do objeto **HttpRequest** da mesma forma que você gravou cookies para o objeto **HttpResponse**. O exemplo de código a seguir mostra duas maneiras para obter o valor de um cookie chamado **usrNome** e exibir seu valor na página:

```
if(Request.Cookies["usrNome"] != null)
   Response.Write(Server.HtmlEncode(Request.Cookies["usrNome"].Value));

if(Request.Cookies["usrNome"] != null)
{
   HttpCookie aCookie = Request.Cookies["usrNome"];
   Response.Write(Server.HtmlEncode(aCookie.Value));
}
```

Lendo Cookies

Antes de tentar obter o valor de um cookie, verifique se o cookie existe; se o cookie não existir, você receberá uma exceção **NullReferenceException**. Observe também que o método **HtmlEncode** foi chamado para codificar o conteúdo de um cookie antes de exibi-lo na página. Isso faria com que um usuário mal-intencionado não adicionasse script executável no cookie.

Ler o valor de uma subchave em um cookie é semelhante a configurá-lo. O exemplo de código a seguir mostra uma maneira para obter o valor de uma subchave:

```
if(Request.Cookies["userInfo"] != null)
{
   Response.Write(
      Server.HtmlEncode(Request.Cookies["usrInfo"]["usrNome"]) + “<br>”);

   Response.Write(
      Server.HtmlEncode(Request.Cookies["usrInfo"][“ultimaVisita"]) + “<br>”);
}
```

Lendo Cookies

As subchaves em um cookie são digitadas como uma coleção de tipo **NameValueCollection**. Portanto, outra maneira para obter uma subchave individual é obter a coleção de subchaves e, em seguida, extrair o valor da subchave pelo nome, conforme mostrado no exemplo o seguir:

```
if(Request.Cookies["userInfo"] != null)
{
   System.Collections.Specialized.NameValueCollection
      UserInfoCookieCollection;

   UserInfoCookieCollection = Request.Cookies["userInfo"].Values;
   Response.Write(
      Server.HtmlEncode(UserInfoCookieCollection["usrNome"]) + “<br>”);
   Response.Write(
      Server.HtmlEncode(UserInfoCookieCollection[“ultimaVisita"]) + “<br>”);
}
```

Excluindo Cookies

Excluir um cookie — fisicamente removê-lo do disco rígido do usuário — é uma variação de modificá-lo. Você não pode remover um cookie diretamente, pois o cookie está no computador do usuário. No entanto, você pode conseguir que o navegador exclua o cookie para você. A técnica é criar um novo cookie com o mesmo nome que o cookie a ser excluído, mas definir a validade do cookie como uma data anterior a hoje. Quando o navegador verifica a validade do cookie, o navegador descarta o cookie agora desatualizado. O exemplo de código a seguir mostra uma maneira para excluir todos os cookies disponíveis para o aplicativo:

```
HttpCookie aCookie;
string cookieNome;
int limit = Request.Cookies.Count;
for (int i=0; i<limit; i++)
{
   cookieNome = Request.Cookies[i].Name;
   aCookie = new HttpCookie(cookieNome);
   aCookie.Expires = DateTime.Now.AddDays(-1);
   Response.Cookies.Add(aCookie);
}
```

Excluindo Cookies

Para excluir uma subchave individual, você manipula a coleção **Values** do cookie, que retém as subchaves. Primeiro, você recria o cookie obtendo-o a partir do objeto **Cookies**. Em seguida, você pode chamar o método **Remove** da coleção **Values**, passando para o método **Remove** o nome da subchave para excluir. Você então adiciona o cookie à coleção **Cookies** para que ele seja enviado em sua forma modificada de volta para o navegador. O exemplo de código a seguir mostra como excluir uma subchave. No exemplo, o nome da subchave a remover é especificado em uma variável.

```
string subkeyNome;
subkeyNome = "usrNome";
HttpCookie aCookie = Request.Cookies["usrInfo"];
aCookie.Values.Remove(subkeyNome);
aCookie.Expires = DateTime.Now.AddDays(1);
Response.Cookies.Add(aCookie);
```

Referências

https://msdn.microsoft.com/pt-br/library/ms178194(v=vs.100).aspx

http://www.devmedia.com.br/configurando-aplicacoes-asp-net-parte-i/2473#ixzz3nwJaj0XB