

# Gerenciamento de estado em ASP.NET

Prof. Felipe Fontoura

# Scripts no cliente

- ▶ São códigos de programação processados pela estação Cliente.
- ▶ Tratam apenas de formatações dinâmicas, pequenas consistências de telas e validações de entrada de dados.
- ▶ São processados por um navegador com capacidade para interpretar a linguagem em que foram escritos.
- ▶ Maiores problemas:
  - Incompatibilidade na interpretação da linguagem utilizada no script entre os navegadores disponíveis no mercado.
  - O código do programador fica exposto ao usuário.

# Scripts no Servidor

- ▶ O uso de linguagens script no computador Cliente não é suficiente para a criação de aplicações voltadas a negócios em sites da Web.
- ▶ Os scripts no lado Cliente fazem o seu papel na camada de apresentação ao usuário (Interface).
- ▶ Porém, a ferramenta usada para tratar a camada de regras de negócio e acessar a camada de dados são os **scripts no lado do Servidor**.

# Scripts no Servidor

- ▶ Um script no lado do Servidor significa que há um código (programa) embutido numa página Web que é executado pelo Servidor quando um computador Cliente faz uma solicitação.
- ▶ Este código normalmente:
  - Obtém dados a partir de fontes de dados (bancos de dados SQL ou arquivos XML);
  - Executa regras de negócio e;
  - Constrói uma página HTML para apresentar os resultados obtidos ao usuário.
- ▶ O resultado final é uma página Web em HTML padrão (compatível com qualquer browser) enviada ao navegador do Cliente.

# Vantagens da programação no Servidor

## ▶ Por que código no servidor?

- Acessibilidade
  - Você pode acessar a Internet de qualquer browser, qualquer dispositivo, a qualquer hora, em qualquer lugar, pois são geradas em HTML padrão
- Gerenciamento
  - Não exige a distribuição de código do aplicativo cliente/servidor
  - Facilidade para mudar o código e atualizar a aplicação
  - O resultado do aplicativo é automaticamente distribuído pela rede, pois o aplicativo “roda” no Servidor

## ▶ Segurança

- O código fonte não é exposto
- Uma vez que o usuário é autenticado, só pode executar certas ações

## ▶ Escalabilidade

- Arquitetura do aplicativo baseada em 3 camadas

# Desvantagens da programação no Servidor

- ▶ Sobrecarga do Servidor quando milhares de usuários solicitarem execuções de funções dos aplicativos.
- ▶ Maior vulnerabilidade quanto a questões de segurança.
- ▶ Dificuldade para controle do “estado da aplicação” para cada usuário.

# Gerenciamento de Estado – Cookies

- ▶ O estado é um termo usado para descrever as informações que devem ser mantidas para descrever o usuário e o que ele está “fazendo” a cada momento quando está utilizando um aplicativo.
- ▶ Sendo o HTTP um protocolo sem estado, os aplicativos Web não têm meios “naturais” de controlar quem é o usuário ou o que ele está fazendo.
- ▶ Os cookies são uma maneira de resolver este problema, e correspondem à uma tecnologia usada para armazenar informações na máquina do usuário.

# Gerenciamento de Estado – Cookies

- ▶ Quando um usuário acessar o site do aplicativo novamente, o navegador enviará ao servidor o cookie salvo anteriormente.
- ▶ Isso permite que o aplicativo personalize conteúdos, recupere informações, ou faça o que mais for necessário para atender cada usuário específico.
- ▶ O aplicativo Web poderá, então, manter o “estado” para cada usuário entre vários acessos subsequentes.
- ▶ Outra maneira de manter o estado ou “rastro” dos usuários é manter suas informações e o que fizeram em um banco de dados (no Servidor).



# Gerenciamento de Estado

- ▶ Cookies são bastante comuns em sites que possuem sistema de permissões e logins. Porém os cookies são armazenados no lado do cliente. Também a gerenciamento de estado do lado servidor.
- ▶ À seguir uma tabela com alguns objetos utilizados para o gerenciamento de estado tanto do lado cliente, quanto do lado servidor.

# Gerenciamento de Estado – Lado do Servidor.

Server-Side Options	Prós	Contras
Application State	Rápido. Compartilhado por todos usuários.	Estado é armazenado um por server em múltiplas configurações de servidor.
Cache Object (Application Scope)	Semelhante a Application mas inclui expiração via Dependencies	Estado é armazenado um por server em múltiplas configurações de servidor.
Session State	Há três possibilidades: in process, out of process, DB-Backed. Pode ser configurado como cookiless.	Pode ser abusado. É pago um custo de serialização quandoo objeto deixa o processo. No processo requer afinidade com Web Server. Configurações cookiless são mais suscetíveis a ataques Hijack (alteração da página inicial)
Database	Estado pode ser acessado por qualquer servidor em um Web farm.	Pague-se um custo de serialização e persistencia quando o objeto abandona o processo. Requer licença de SQL server.

# Gerenciamento de Estado – Lado do Cliente.

Server-Side Options	Prós	Contras
Cookie	Simples	Pode ser rejeitado pelo browser. Não é apropriado para grandes volumes de dados. Inapropriado para dados sensíveis. Paga-se o custo do tamanho (size) a cada HTTP Request e HTTP Response.
Hidden Field	Simples para dados da página do escopo.	Não é apropriado para grandes volumes de dados e para dados sensíveis.
ViewState	Simples para dados da página do escopo.	Objeto serializado como dado binário Base64 acrescenta aproximadamente 30% de overhead. Pequeno custo de serialização. Tem reputação negativa em particular com DataGrids.
ControlState	Simples para um controle específico de dados da página do escopo.	Como o ViewState, mas usado com controles que requerem ViewState mesmo que o desenvolvedor o tenha desabilitado.
QueryString(URL)	Muito simples e conveniente se você deseja que o usuário modifique diretamente as URLs.	Comparativamente complexo. Não pode lidar com grandes volumes de informação. Inapropriado para dados sensíveis. Facilmente modificado pelo usuário final.

# Gerenciamento de estado – Sessão

- ▶ O objeto Session armazena informações sobre, ou alterar configurações de uma sessão do usuário.
- ▶ Variáveis armazenadas em um objeto de sessão armazenam informações sobre um único usuário, e estão disponíveis para todas as páginas em um aplicativo.
- ▶ O servidor cria um novo objeto de sessão para cada novo usuário, e destrói o objeto de sessão quando a sessão expira.
- ▶ Uma sessão começa quando:
  - Um usuário acessa um web site.
- ▶ A sessão termina se:
  - Um usuário não solicitou ou atualizou uma página na aplicação por um período determinado. Por padrão: 20 minutos.
  - O usuário deixa a página por um método de *log out*

# Gerenciamento de estado – Sessão

- ▶ As sessions são armazenadas na memória do servidor: pode-se sobrecarregar o servidor e se ele precisar de recursos vai matar suas sessions, e seu dados somem.
- ▶ As sessions expiram como tempo. O tempo é configurado no servidor. Você pode definir o tempo no código mas nem todos os servidores acatam.

```
protected void btnEnviar_Click(object sender, EventArgs e)
{
    //Armazenando valores dos textbox em session
    Session["nome"] = txtNome.Text;
    Session["site"] = txtSite.Text;

    //Redirecionando a aplicação para o form2.
    Response.Redirect("form2.aspx");
}
```

# Gerenciamento de estado – Application

- ▶ Equivalente a uma variável global no ASP.NET
- ▶ Cada servidor possui o seu próprio objeto Application
- ▶ Cuidado ao usar num contexto de Web farm.
- ▶ Como as aplicações ASP.NET são multithreads acesso aos objetos Application devem ser usados com
  - Application.Lock();
  - Application.Unlock();

```
Application.Lock();  
Application["GlobalCount"] = (int)Application["GlobalCount"] + 1;  
Application.Unlock();
```

# Gerenciamento de estado – QueryStrings

- ▶ Lugar para dados de navegação e não de usuário.
- ▶ É o elemento mais vulnerável a ataques de Hacker
- ▶ Você precisa se preparar para lidar com páginas não encontradas (erro 404)
  - `Response.StatusCode = 404;`
  - É a primeira informação a ser lida, mesmo antes da página ser carregada já esta na barra de endereços.
- ▶ Exemplos:
  - `http://reviews.cnet.com/Philips\_42PF9996/4505-6482\_7-31081946.html?tag=cnetfd.sd`
  - `http://www.hanselman.com/blog/CategoryView.aspx?category=Movies`

# Gerenciamento de estado – A Escolha

- Quanta informação precisa ser armazenada?
- O browser do cliente aceita cookies?
- Você pretende armazenar a informação no cliente ou no servidor?
- A informação é sensível? Precisa ser protegida?
- Questões relativas à performance e a largura de banda são importantes?
- Você precisa armazenar informações para cada usuário?
- Quanto tempo a informação precisa ficar armazenada?
- A informação precisa ser compartilhada entre servidores ou entre processos?



# **Criando Cookies em ASP.NET**

Ana Paula Citro Fugarra

Cookie é uma pequena quantidade de texto enviada junto com a requisição do usuário. Cada web site grava e armazena os seus cookies separadamente no computador do usuário em pequenos arquivos-texto ou na memória, numa sessão do browser do cliente.

Um cookie pode ser temporário, com data e hora para expirar, ou persistente.

Quando o usuário requisita uma página de um web site que previamente gravou um cookie, este é enviado junto com a requisição do usuário e lido pela aplicação web sempre que o usuário visita o web site.

Os cookies ajudam a armazenar informações sobre os visitantes, mas eles contêm uma série de limitações, como, por exemplo: armazenam informações com no máximo 4096 bytes de tamanho. Além disso alguns browsers limitam o número de cookies que podem ser armazenados de cada vez. E para piorar a situação, outros simplesmente não os aceitam.

# Gravando Cookies

Cookies são armazenados no cliente pelo objeto **HttpResponse**, o qual possui a propriedade **Cookies**, ou são armazenados pela classe **HttpCookie**.

Cada cookie deve ter um nome único e um valor.

Como os cookies são identificados pelo nome, gravar um cookie com um nome que já existe faz com que o cookie existente seja sobrescrito.

Se não definimos a data na qual o cookie expira, ele é criado, mas não é armazenado no HD do usuário. Neste caso, o cookie é descartado quando o usuário fecha o browser.

# Gravando Cookies de valor único

Usando a propriedade **Cookies**:

```
Response.Cookies["NomeCookie"].Value = "Alfredo";  
Response.Cookies["NomeCookie"].Expires = DateTime.Now.AddDays(1d);
```

Usando uma instância da classe **HttpCookie**:

```
HttpCookie cook = new HttpCookie("NomeCookie");  
cook.Value = "Alfredo";  
cook.Expires = DateTime.Now.AddDays(1d);  
Response.Cookies.Add(cook);
```

# Gravando Cookies com múltiplos valores

Para gravarmos múltiplos valores em um cookie, basta combinar o nome do cookie com subchaves.

Usando a propriedade **Cookies**:

```
Response.Cookies["NomeCookie"]["NomeValor"] = "Alfredo";  
Response.Cookies["NomeCookie"]["NomeValor"] = "ASP.NET com C#";  
Response.Cookies["NomeCookie"].Expires = DateTime.Now.AddDays(1d);
```

Usando uma instância da classe **HttpCookie**:

```
HttpCookie cook = new HttpCookie("NomeCookie");  
cook["NomeValor"] = "Alfredo";  
cook["NomeValor"] = "ASP.NET com C#";  
cook.Expires = DateTime.Now.AddDays(1d);  
Response.Cookies.Add(cook);
```

# Restringindo o acesso a cookies

Por padrão, um cookie está acessível para qualquer página do web site que o gravou, mas é possível limitar o escopo do cookie, ou seja, podemos definir quais páginas podem ler determinado cookie.

A propriedade **Path** define o diretório da aplicação onde o cookie está acessível. Por exemplo, se um web site possui a URL <http://www.meusite.com.br>, e a propriedade **Path** foi definida como “teste”

```
Response.Cookies["NomeCookie"].Value = "Alfredo";  
Response.Cookies["NomeCookie"].Expires = DateTime.Now.AddDays(1d);  
Response.Cookies["NomeCookie"].Path = "/teste";
```

somente páginas do diretório-teste podem acessar o cookie.

**<http://localhost:1192/WebSite1/teste/Default.aspx>**

**<http://www.meusite.com.br/teste/Default.aspx>**

# Lendo Cookies

Para ler o conteúdo de um cookie usamos o objeto **HttpRequest**. O cookie é acessado pelo nome e seu valor retornado pela propriedade **value**. Antes de acessar um cookie, verifique se ele existe; senão, uma exceção do tipo **NullReferenceException** é retornada.

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

A classe **HttpCookie** também pode ser empregada na leitura de cookies.

```
if ( Request.Cookies["NomeCookie"] != null ) {  
    HttpCookie cook = Request.Cookies["NomeCookie"] ;  
    Response.Write (Server.HtmlEnconde (cook.Value)) ;  
}
```

# Lendo Cookies com múltiplos valores

O cookie gravado como

```
Response.Cookies["Livro"]["Autor"] = "Alfredo";
```

Pode ser lido usando:

```
if( Request.Cookies["Livro"] != null ){  
    if( Request.Cookies["Livro"]["Autor"] != null ){  
        Response.Write(Server.HtmlEncode(Request.Cookies["Livro"]["Autor"]));  
    }  
}
```



# Modificando e excluindo cookies

Na verdade, não existe uma forma de modificar um cookie, mas podemos recriá-lo e atribuir-lhe novos valores. Exemplo:

Cookie já criado:

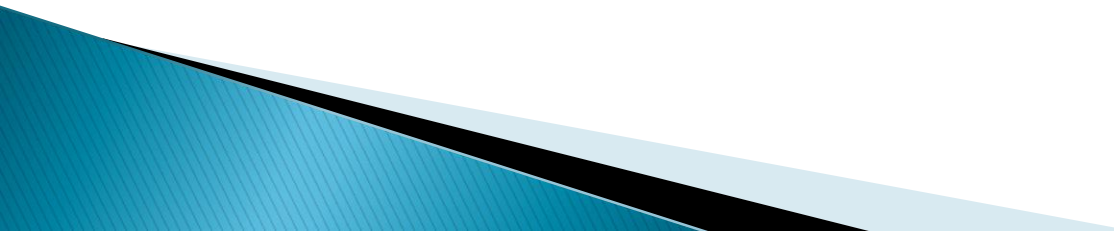
```
Response.Cookies["Livro"]["Titulo"] = "ASP.NET com C#";  
Response.Cookies["Livro"].Expires = DateTime.Now.AddDays(2d);
```

Cookie alterado, recriado:

```
Response.Cookies["Livro"]["Titulo"] = "Java - JSP";  
Response.Cookies["Livro"].Expires = DateTime.Now.AddDays(1d);
```

Para excluir um cookie basta definir a propriedade **Expires** com um valor negativo.

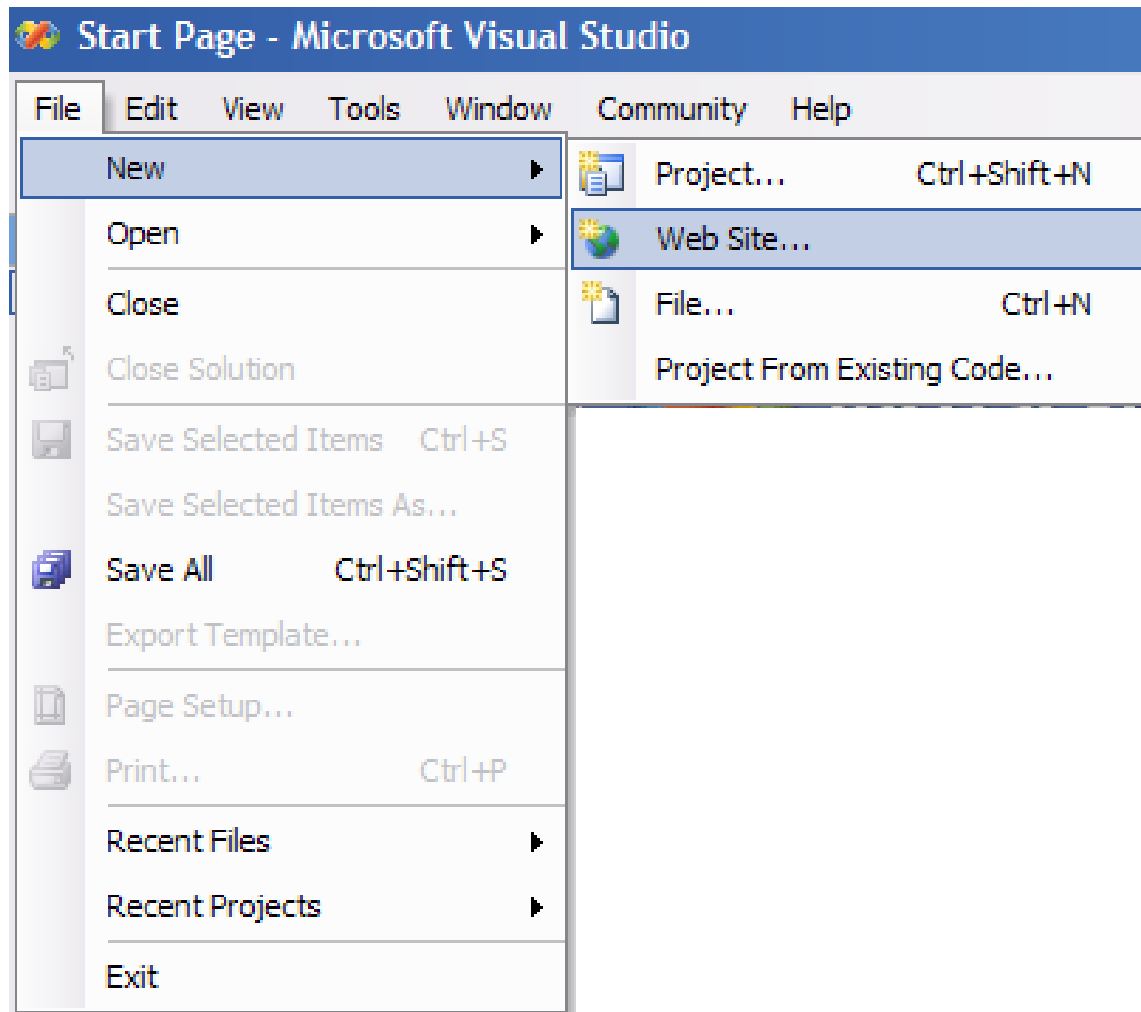
```
Response.Cookies["Livro"].Expires = DateTime.Now.AddDays(-1d);
```



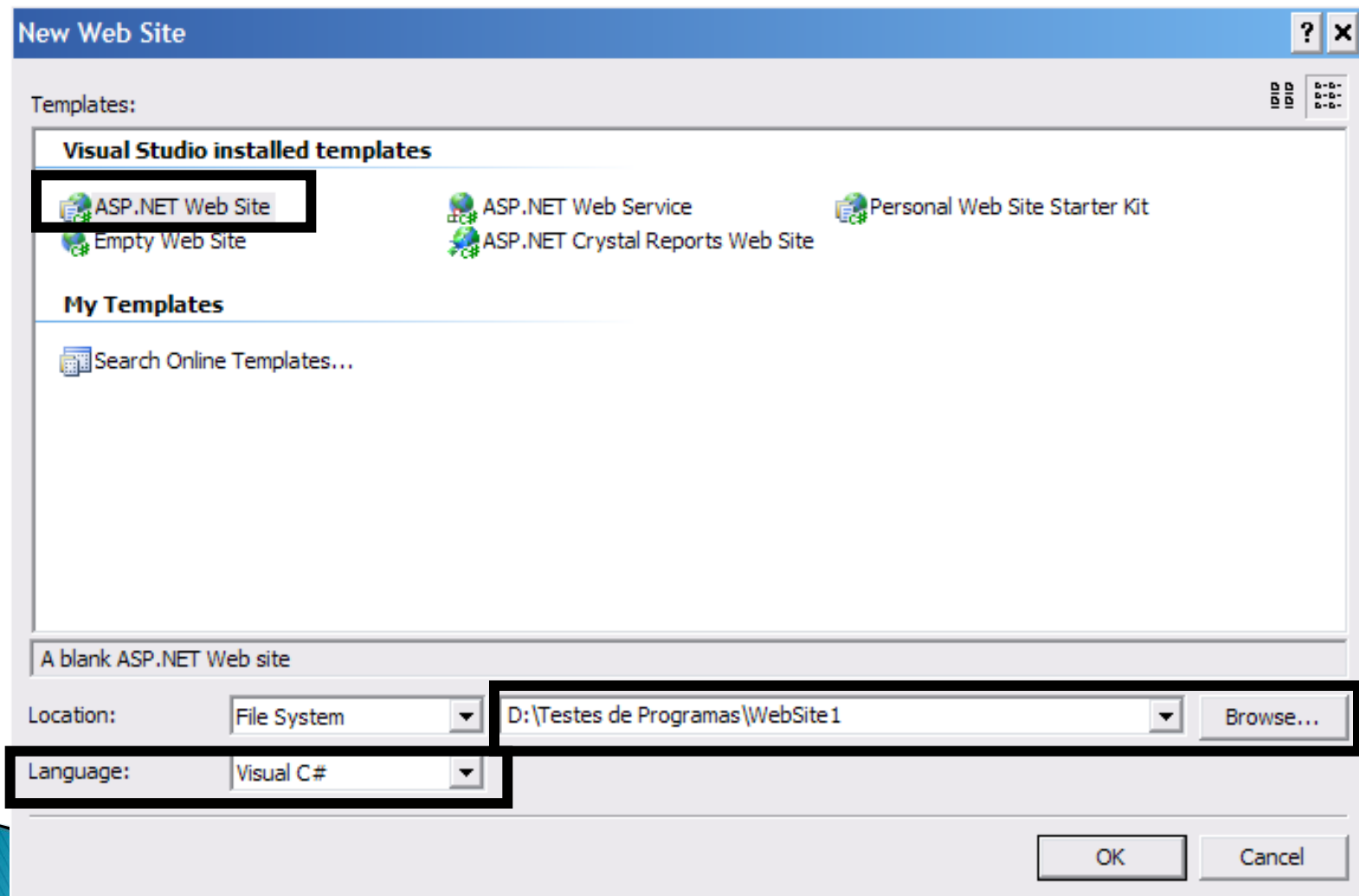
# Exemplo:

# Cookies em ASP.NET

## 1º Passo: Criar um Web Site



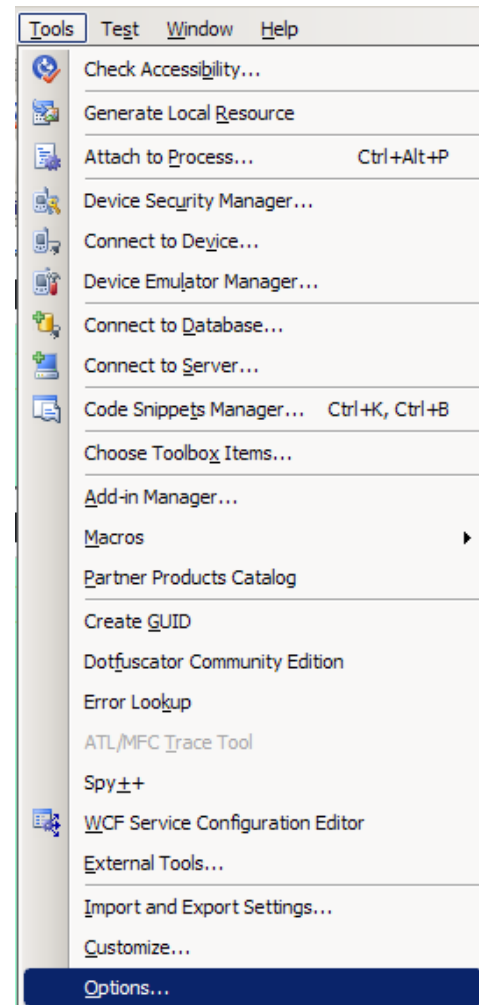
## 2º Passo: Selecionar ASP.NET Web Site

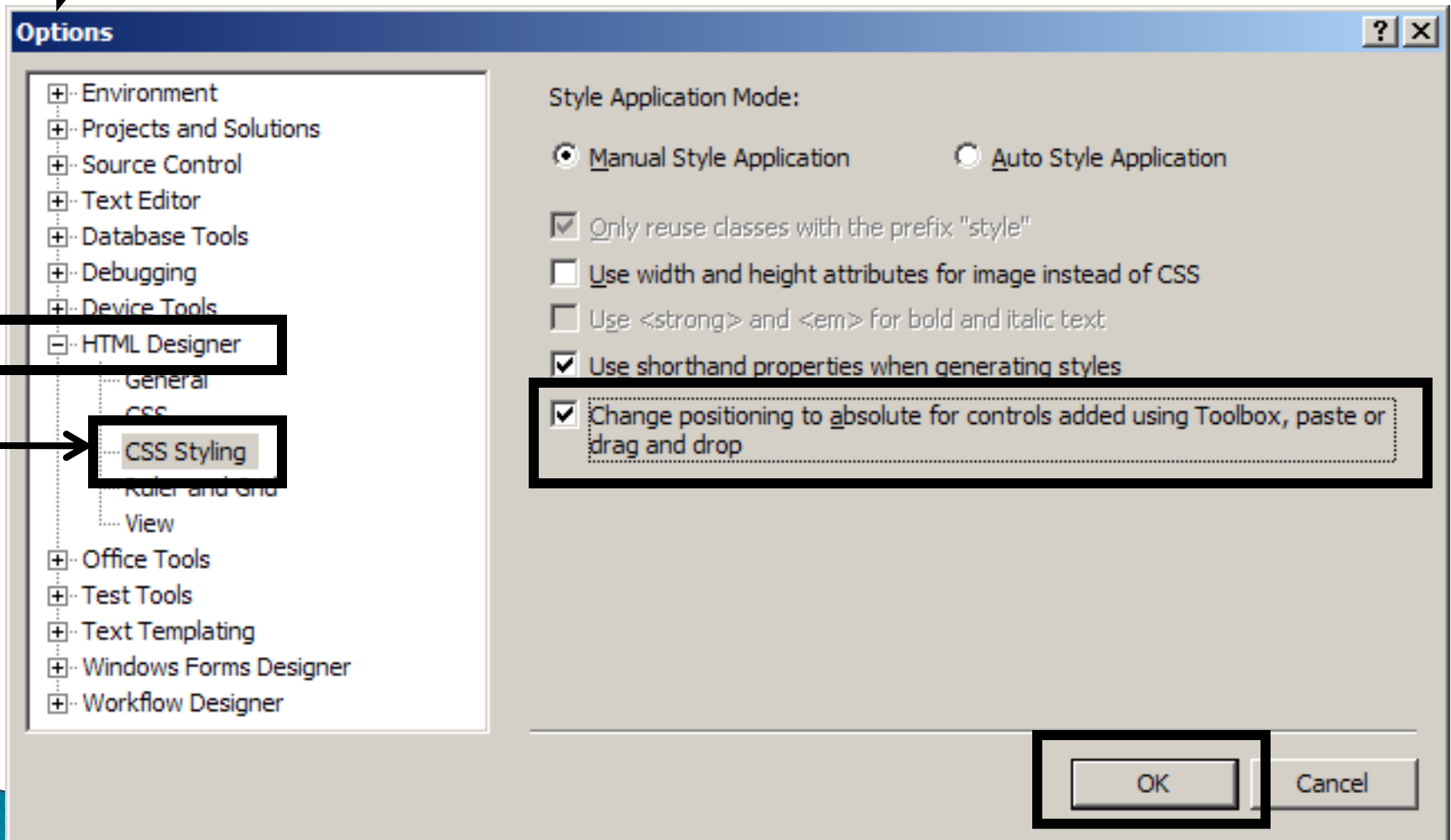


**3º Passo:** Selecionar no rodapé da página a opção



**4º Passo:** Alterar o layout da página





**5º Passo:** Construir a página conforme a figura abaixo:

Start Page Default.aspx.cs **Default.aspx**

Autor:  Gravar Cookie

Título:  Ler Cookie


lblSaidaAutor Matar Cookie

lblSaidaTitulo

lblMensagem

ID	Tipo	Text	ForeColor
lblAutor	Label	Autor:	
txtNome	TextBox		
btnGravar	Button	Gravar cookie	
lblTitulo	Label	Título:	
txtTitulo	TextBox		
btnLer	Button	Ler cookie	
btnMatar	Button	Matar cookie	
lblSaidaAutor	Label		
lblSaidaTitulo	Label		
lblMensagem	Label		Red


**6º Passo:** Acione o Click no **btnGravar** e escreva o código abaixo:



```
protected void btnGravar_Click(object sender, EventArgs e)
{
    if (!string.IsNullOrEmpty(txtNome.Text.Trim()) &&
        !string.IsNullOrEmpty(txtTitulo.Text.Trim()))
    {
        Response.Cookies["Livro"]["Autor"] = txtNome.Text;
        Response.Cookies["Livro"]["Titulo"] = txtTitulo.Text;
        Response.Cookies["Livro"].Expires = DateTime.Now.AddDays(2);
        lblMensagem.Text = string.Empty;
    }
    else
    {
        lblMensagem.Text = "Preencha todos os campos do formulário.";
        txtNome.Focus();
    }
}
```



7º Passo: Acione o Click no **btnLer** e escreva o código abaixo:



```
protected void btnLer_Click(object sender, EventArgs e)
{
    if (Request.Cookies["Livro"] != null)
    {
        if (Request.Cookies["Livro"]["Autor"] != null &&
            Request.Cookies["Livro"]["Titulo"] != null)
        {
            lblSaidaAutor.Text = "Autor: " +
                Server.HtmlEncode(Request.Cookies["Livro"]["Autor"]);
            lblSaidaTitulo.Text = "Livro: " +
                Server.HtmlEncode(Request.Cookies["Livro"]["Titulo"]);
        }
    }
}
```

**8º Passo:** Acione o Click no **btnMatar** e escreva o código abaixo:

```
protected void btnMatar_Click(object sender, EventArgs e)
{
    Response.Cookies["Livro"].Expires = DateTime.Now.AddDays(-1d);
}
```

**9º Passo:** Compile e teste seu programa.

Procure o cookie em sua máquina, ele será criado na pasta Cookies, do usuário logado, na pasta Documents and Settings no C:\

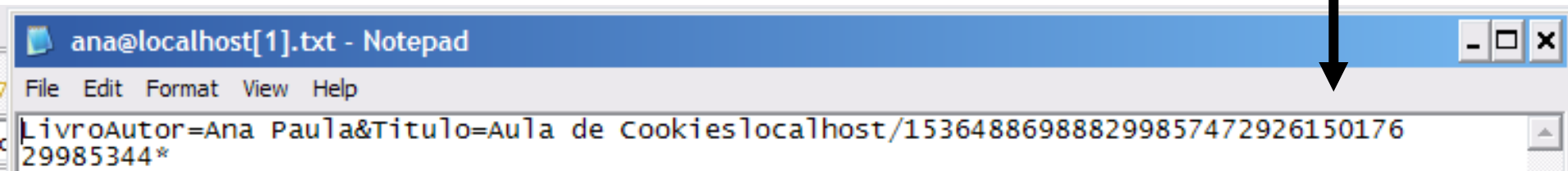
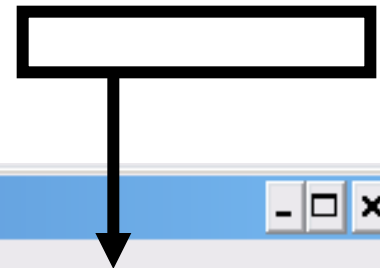
Autor:

Título:

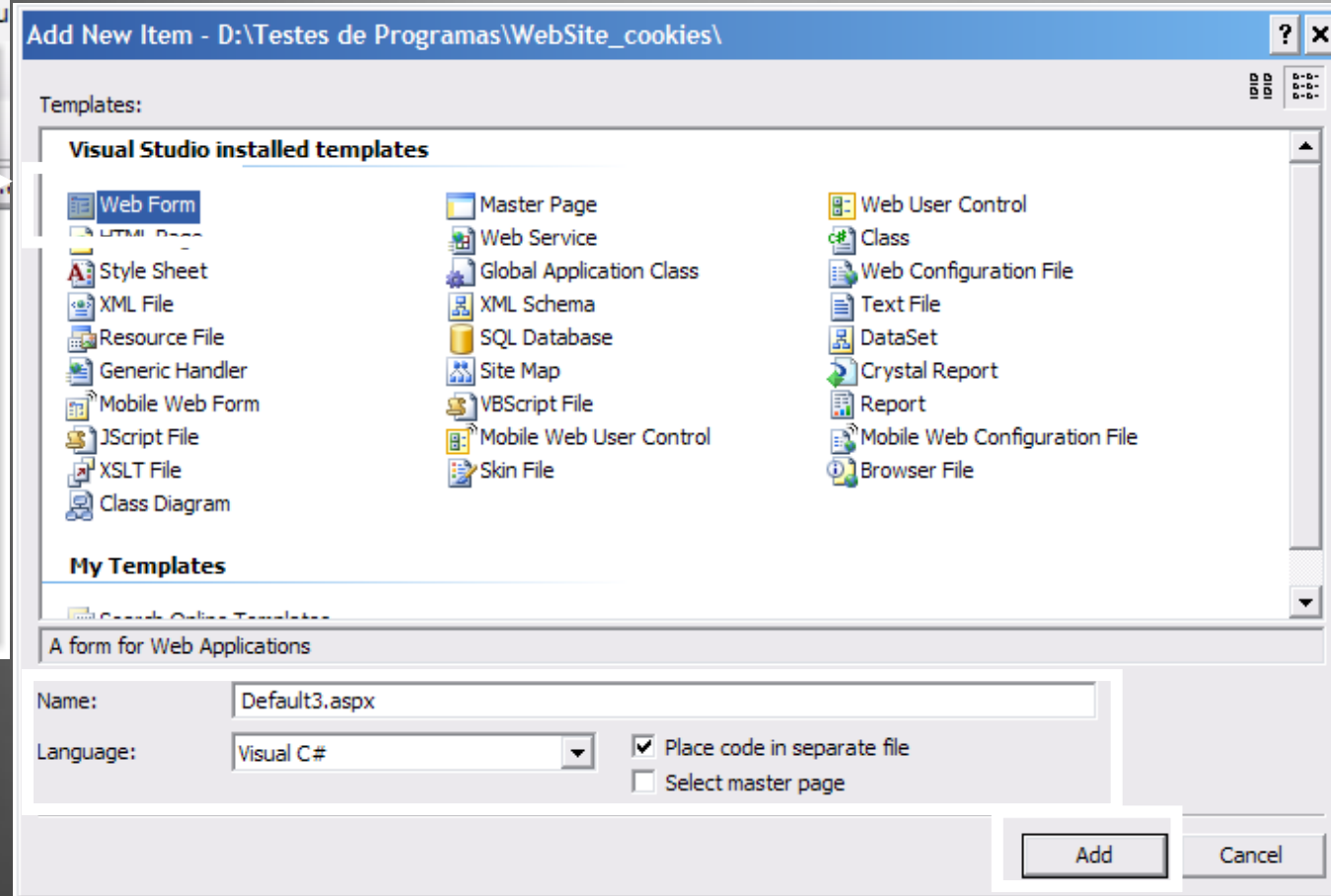
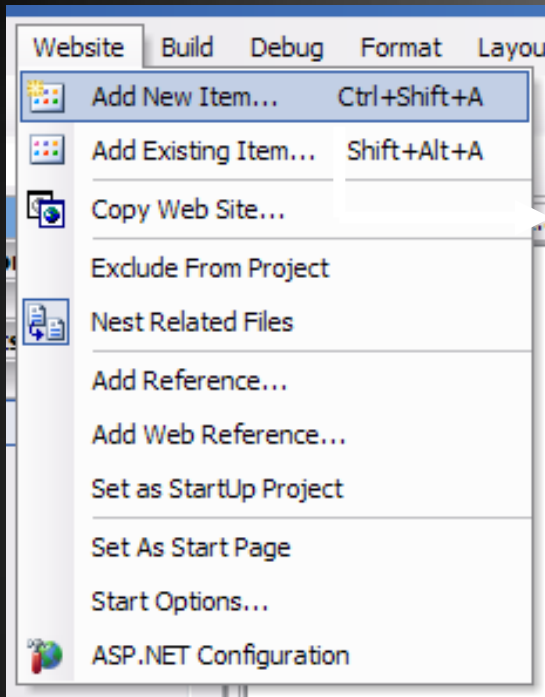
Gravar Cookie

Ler Cookie

Matar Cookie



# Multi Web Forms



Para abrir os outros Web Forms escreva o código abaixo:

```
Response.Redirect ("NomedPagina.aspx");
```

# Exercício

- ▶ **Questão 01.** O formulário a seguir é um formulário padrão para comentários em Blogs. Construa conforme abaixo uma página e **em outra página** (após click em *Enviar Comentário*) mostre os dados inseridos:

## Deixe um comentário

Nome:

E-mail:

Website:

Comentário:

Observação: Seu comentário será publicado após passar pelo moderador

Enviar Comentário