



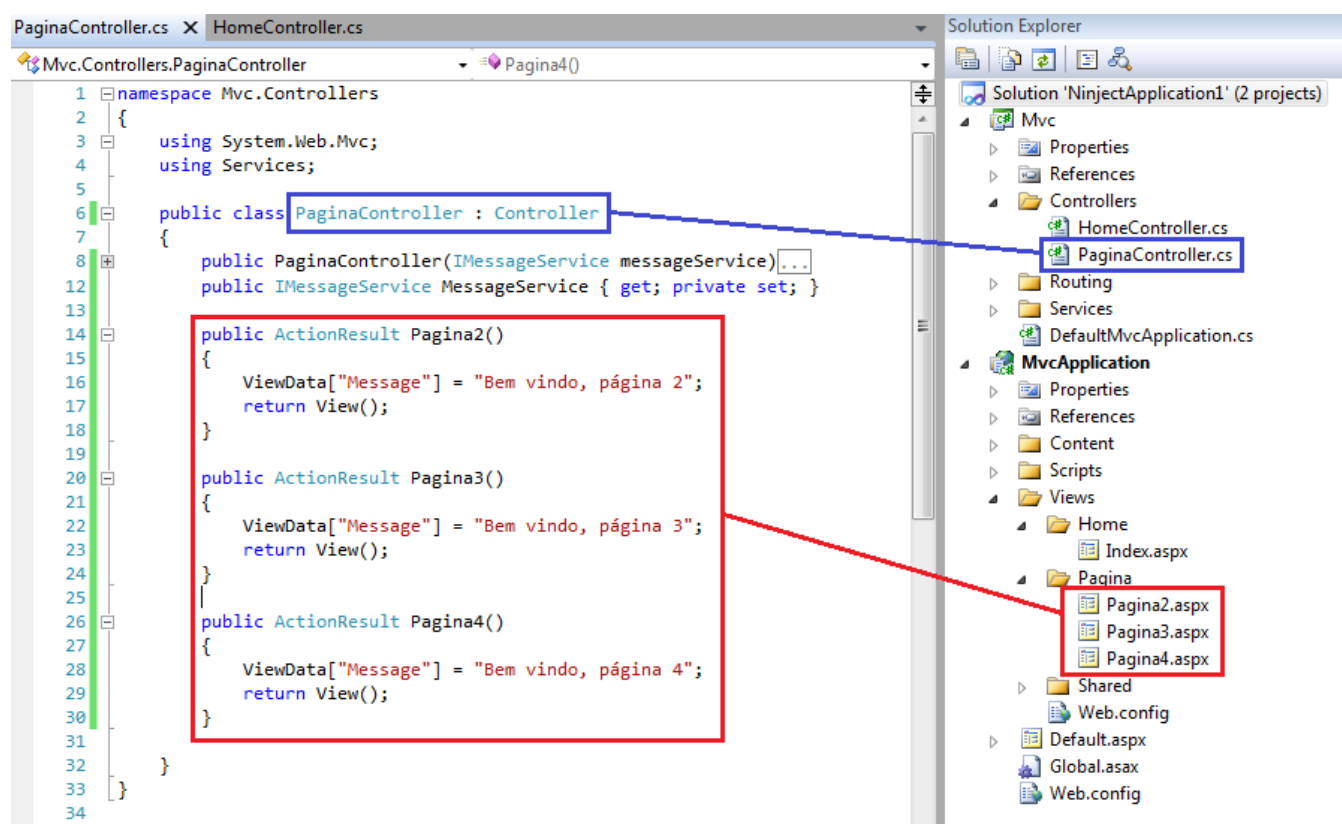
post favorito comentários

ASP.NET MVC – Autenticação básica

Neste artigo, demonstrarei como podemos restringir o acesso aos nossos Controllers/Views de modo que, apenas os usuários que forem autenticados no sistema possam acessar tais recursos.

 0 Curtir 1 Gostei (1)  (0)

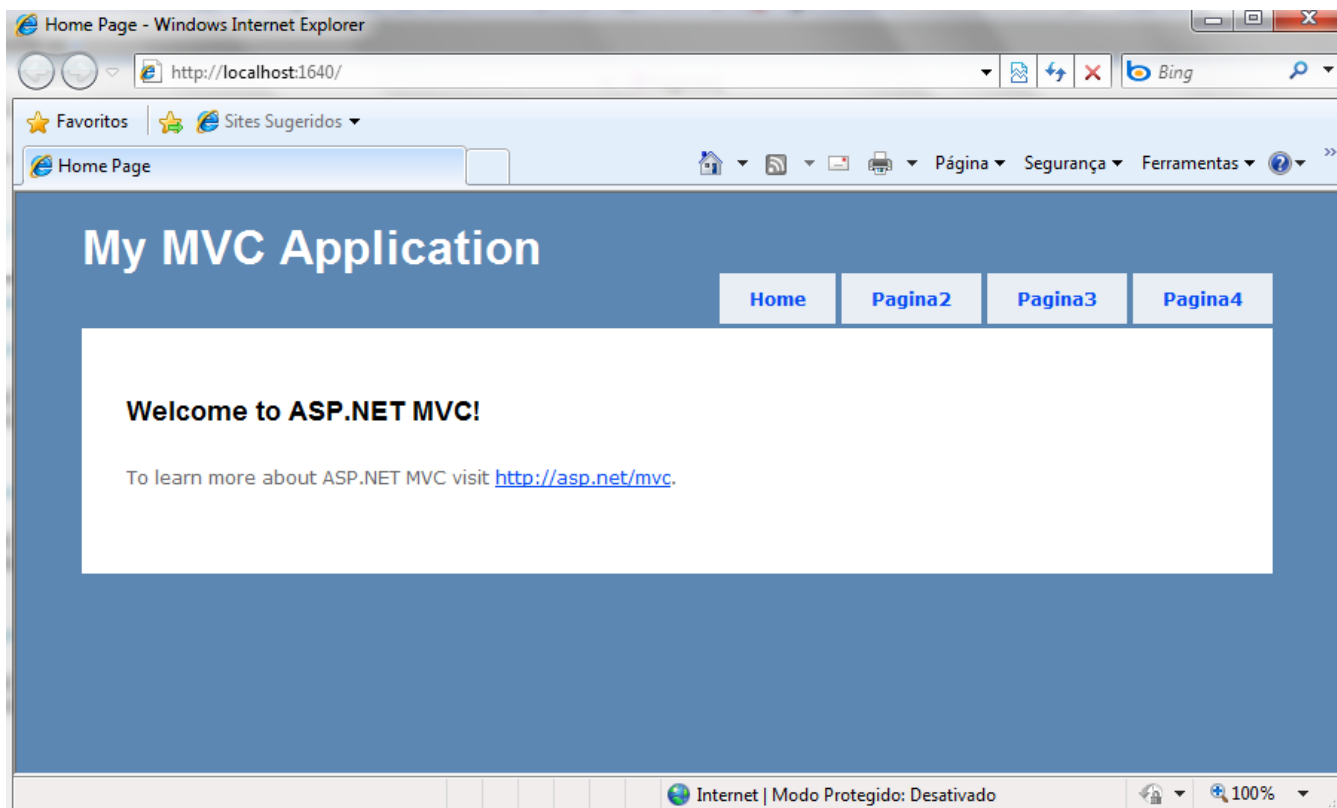
Vamos imaginar o projeto com a seguinte estrutura:



Na imagem acima podemos observar uma estrutura básica de um projeto em MVC, como contornado na figura, possuímos um Controller chamado *PaginaController*, que possui 3 métodos que retornam uma *ActionResult* (*Pagina2*, *Pagina3* e *Pagina4*).

OBS: Gostou da organização de nosso projeto? Veja como deixá-lo assim [aqui](#).

Ao executar nossa aplicação, o resultado apresentado é o seguinte:



Até aqui tudo tranquilo, porém gostaríamos que apenas os usuários autenticados no sistema pudessem acessar tais menus.

Para tanto, vamos decorar nosso Controller chamado *PaginaController* com o atributo `AuthorizeAttribute`.

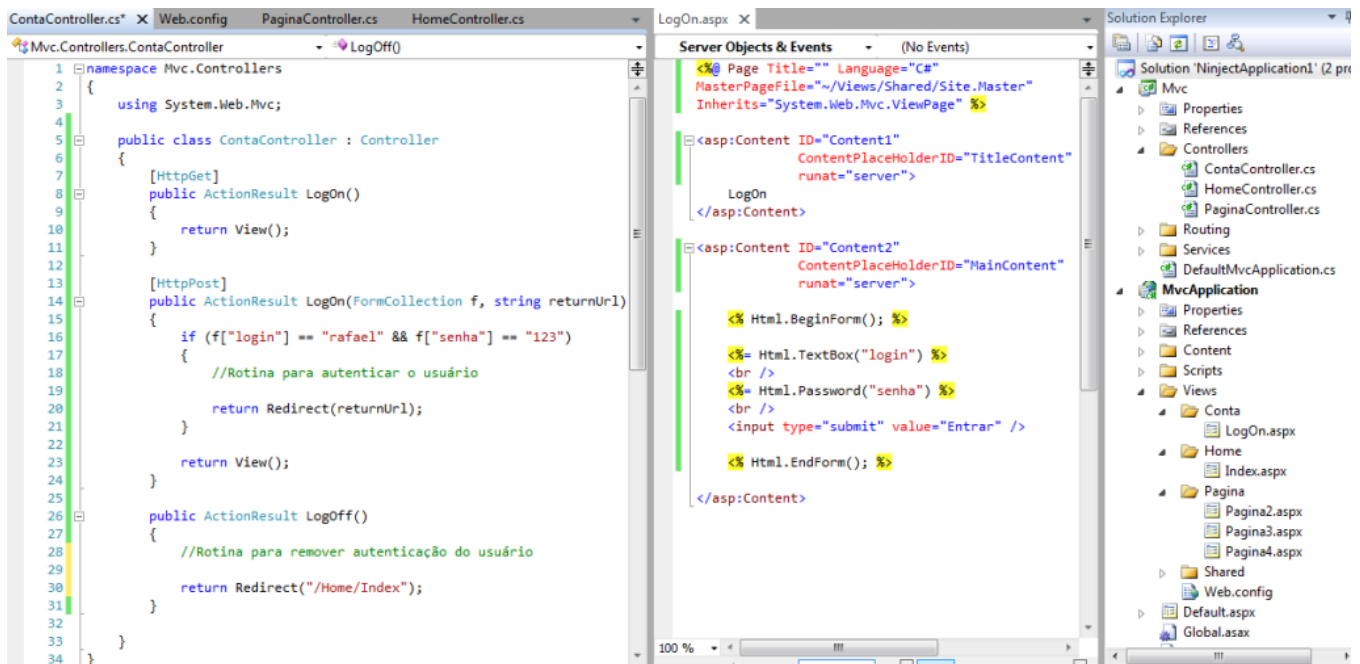
```
[Authorize]
public class PaginaController : Controller
{
    public PaginaController(IMessageSe
```

Isto define que **TODOS** os métodos existentes dentro de *PaginaController* necessitam da autenticação por parte do usuário. Podemos também, defini-lo apenas em alguns métodos ao invés de definirmos no Controller.

Agora, vamos definir o seguinte código em nosso arquivo web.config:

```
<authentication mode="Forms">
  <forms loginUrl="~/Conta/LogOn" />
</authentication>
```

Definimos acima que, a autenticação do usuário será feita através de um formulário, e este formulário foi definido como `/Conta/LogOn`. Devemos então, criar um Controller *Conta* com o método `LogOn`. Ao final nosso projeto deve estar como abaixo:



Acima podemos ver que criamos um Controller chamado ContaController, que se responsabiliza apenas por efetuar o *Logon* e *Logoff* do usuário.

Esclarecendo como “as coisas funcionam”, quando for realizada uma requisição a qualquer *ActionResult* que possua o atributo *Authorize*, como por exemplo o método *Pagina2* dentro de *PaginaController*, é verificado se o usuário está autenticado no sistema, se a resposta for positiva é redirecionado para sua requisição solicitada, ou seja, */Pagina/Pagina2*.

Caso o usuário não esteja autenticado, o mesmo é redirecionado para o formulário de autenticação definido no web.config (*/Conta/LogOn*). Note que, na imagem acima, ao realizar o POST do formulário, é verificado se o login e senha do usuário são respectivamente “rafael” e “123”, para a realização da rotina de autenticação.

Vamos então criar a autenticação do usuário, a forma mais simples é utilizarmos a classe [FormsAuthentication](#) localizada dentro no namespace *System.Web.Security*, como demonstrado abaixo:

```
[HttpPost]
public ActionResult LogOn(FormCollection f, string returnUrl)
{
    if (f["login"] == "rafael" && f["senha"] == "123")
    {
        //Rotina para autenticar o usuário
        System.Web.Security.FormsAuthentication.SetAuthCookie(f["login"], false);
        return Redirect(returnUrl);
    }
    return View();
}

public ActionResult LogOff()
{
    //Rotina para remover autenticação do usuário
    System.Web.Security.FormsAuthentication.SignOut();
    return Redirect("/Home/Index");
}
```

Observe, utilizamos a rotina [SetAuthCookie](#) para criarmos um cookie de autenticação, este método recebe 2 parâmetros: o primeiro é o nome do cookie, e o segundo é se o cookie será persistente, ou seja, mesmo fechando o browser o usuário continua autenticado.

Note também que foi recebido um parâmetro chamado returnUrl, este parâmetro nos mostra qual a requisição que o usuário solicitou, para que seja possível redirecioná-lo após sua autenticação.

Para efetuarmos o Logoff foi utilizado o método [SignOut](#), responsável por remover a configuração de autenticação do usuário.

PRONTO! agora já é possível solicitar a autenticação do usuário quando desejado. Vamos apenas criar uma verificação em nossa Master Page chamada Site.Master para conferir se o usuário esta autenticado ou não, afim de exibirmos um link de LogOn e LogOff, conforme abaixo:

```

<div id="divLogin">
  <% if (HttpContext.Current.User.Identity.IsAuthenticated)
  { %>
    Bem vindo <%= HttpContext.Current.User.Identity.Name %> <a href="/Conta/LogOff">Sair</a>
  <%}
  else
  { %>
    <a href="/Conta/LogOn">Entrar</a>
  <%} %>
</div>

```

Verifica se o usuário está autenticado

Obtém o nome do cookie

**Rafael Zaccanini**

Graduado em Análise e Desenvolvimento de Sistemas pela Veris-IBTA. Analista de Sistemas especialista na plataforma Microsoft .NET, atualmente desenvolve projetos web com a arquitetura MVC. Possui experiência com as tecnologias SQL [...]

0 Curtir 1

Gostei (1) (0)

COMENTE TAMBÉM

0 COMENTÁRIO

Nenhum comentário foi postado - seja o primeiro a comentar!

+.net

Publicidade

VAMOS CRIAR APPS

QUALCOMM

Developer Network

► CONHEÇA AS FERRAMENTAS

Serviços

- Inclua um comentário
- Adicionar aos Favoritos
- Marcar como lido/assistido
- Incluir anotação pessoal
- Versão para impressão

Mais posts

Artigo

Gerenciando exceções com C#

Revista

Revista .net Magazine 108

Video aula

SRP - Single Responsibility Principle - Curso Padrões de Projeto com C# - 52

Video aula

Interpreter - Curso Padrões de Projeto com C# - 51

Video aula

Usando um Object Pool para Conexões - Curso Padrões de Projeto com C# - 50

Video aula

Criando um Object Pool Genérico - Curso Padrões de Projeto com C# - 49

Video aula

Private Class Data - Curso Padrões de Projeto com C# - 48

Video aula

Códigos de Barras - Curso de Crystal Reports para Visual Studio - Aula 23

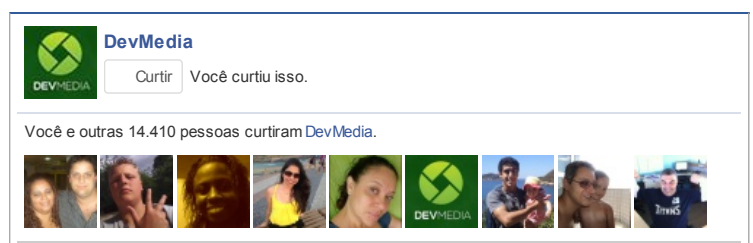
Video aula

Gráficos - Curso de Crystal Reports para Visual Studio - Aula 22

Video aula

Etiquetas de Mala Direta - Curso de Crystal Reports para Visual Studio - Aula 21

[Listar mais conteúdo](#)





DevMedia | Anuncie | Fale conosco

Hospedagem web por Porta 80 Web Hosting

2013 - Todos os Direitos Reservados a **web-03**