Realizando Autenticação (Authentication) e Autorização (Authorization) em Asp.MVC 1

Olá pessoal!

Nos últimos posts falei basicamente sobre Asp.Net MVC, que, **na minha opinião**, é a melhor forma de desenvolver para Web em Asp.Net. Entenda que com "na minha opinião" eu quero dizer que eu me sinto mais confortável utilizando MVC, mas, de uma forma ou de outra, tudo que é feito em MVC pode ser feito em WebForm ou em qualquer outra tecnologia.

Em todos os posts abordei assuntos básicos, pensando numa linha de o que uma pessoa iniciante precisa saber para conseguir fazer uma página inteira em Asp.Net MVC. Acredito que este é o último post necessário para uma pessoa conseguir fazer um site ou um aplicativo web totalmente em Asp.Net MVC.

Naturalmente, na construção de um site, muitas outras coisas são utilizadas, mas o que foi passado aqui já é o suficiente para fazer um funcional.

Como o título do post fala, falarei da parte de segurança de um site.

Qual é a diferença entre Autenticação e Autorização?

Para falar de segurança, é importante começar por conceitos básicos. Por isso vou explicar a diferença entre as duas palavras mais utilizadas nesse assunto: Autenticação e Autorização .

Autenticação

Basicamente, autenticação é a forma do sistema saber se a pessoa é realmente quem ela diz ser. Por exemplo, para acessar um prédio é normal que o segurança peça um documento oficial com foto, para ver se você é realmente a pessoa que diz ser. Nesse processo, ele está realizando a sua autenticação, autenticando o que você diz ser, com o que o documento oficial informa. Se as duas informações baterem, você está autenticado.

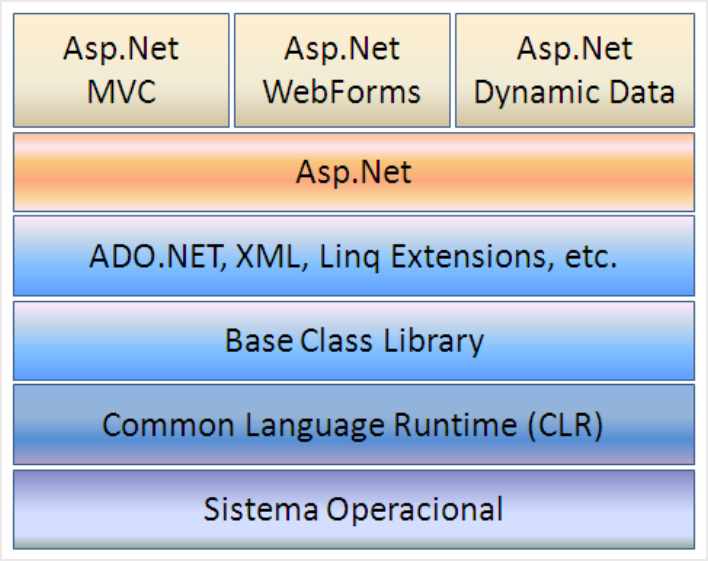
Autorização

Tomando como ponto de partida o exemplo anterior, o fato de você realmente ser quem você diz que é, não te garante acesso ao prédio. Imaginemos que você esteja querendo entrar na sede da Microsoft, o segurança já sabe quem você é, e te pergunta: "*Com que você gostaria de falar?*", e você: "*Steve Ballmer* ", o segurança vai responder para você, depois de fazer uma ligação: "*Desculpe, mas o sr. Steve não está disponível*", isso se ele for educado, mas na prática ele está te falando: "*Você não tem **autorização** para falar com o chefão*". Autorização é basicamente isso: Depois de saber quem você é (autenticação), saber se você pode fazer o que está querendo fazer (autorização).

Como isso é feito em Asp.Net MVC

Para quem já programa em WebForm, ai vai a boa notícia: Praticamente nada muda!

Em um dos post introdutórios ao Asp.Net, utilizei a figura abaixo para exemplificar como é organizado o Asp.Net:



A parte de autenticação em Asp.Net MVC não muda muito do WebForm porque toda a estrutura de

Sobre mim

Meu nome é Frederico Batista Emídio, trabalho com desenvolvimento de sistemas profissionalmente a oito anos, porém, já faço sites pessoais a pelo menos dez anos.
Para saber mais clique aqui.

Listagem por Mês

- 2013
  - fevereiro (1)
  - janeiro (3)
- 2012
  - outubro (1)
  - fevereiro (1)
  - janeiro (1)
- 2011
  - outubro (1)
  - setembro (2)
  - agosto (2)
  - julho (4)
  - maio (4)
  - março (1)
  - fevereiro (1)
  - janeiro (3)
- 2010
  - dezembro (1)
  - novembro (3)
  - outubro (5)
  - setembro (3)
  - agosto (4)
  - julho (1)

Páginas

- Sobre mim.

Blogs que Leio

- Steve Sanderson's blog
  - Knockout 3.0 Release Ca...
  - Knockout v2.3.0 release...
  - Knockout-ES5: a plugin...
- ScottGu's Blog
  - Windows Azure: Announci...
  - Announcing the Release...
  - Windows Azure: New Virt...

Download OPML file

Calendário

<<	Outubro 2013					>>
Se	Te	Qu	Qu	Se	Sá	Do
30	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

Visualizar posts em um Calendário

Busca

Insira um texto Buscar

Incluir comentários na pesquisa

segurança do Asp.Net está no nível ***Asp.Net*** da figura, portanto, todas as tecnologias "finais" da plataforma compartilham a mesma infraestrutura de segurança.

No Asp.Net, existem duas formas de realizar a autenticação/autorização:

Windows Authentication: É a forma de utilizar a autenticação do Windows integrado com o IIS. As credenciais utilizadas são as mesmas credenciais do Active Directory (AD) do servidor Web, que utiliza o protocolo LDAP. Caso o site/sistema esteja sendo acessado de uma intranet, o usuário pode já entrar no site autenticado, de acordo com o usuário logado na máquina do cliente. Caso seja um acesso feito da internet, onde o usuário logado na máquina do cliente não está cadastrado no mesmo AD configurado no servidor, então o browser geralmente exibe uma janela popup solicitando o usuário, senha e domínio no momento em que o usuário acessa o site, para o acesso ser autenticado diretamente do AD do servidor, permitindo assim que apenas usuário autenticados consigam acessar o site.

Forms Authentication: Com Forms Authentication, a responsabilidade de identificar quem está acessando o site fica a cargo do próprio site. Nesse processo você configura uma página responsável por obter as credenciais do acesso (Usuário/Senha). Onde o usuário e senha serão validados fica por responsabilidade do desenvolvedor, pode ser feito em qualquer lugar, como Banco de Dados, arquivos, ou mesmo no AD do servidor.

Nos exemplo vou utilizar Forms Autentication, por ser mais comum de ser utilizado. Além do que eu vou mostrar, existem muitas formas estender ou customizar a arquitetura de segurança do Asp.Net, mas isso é assunto para um outro post, nesse mostraremos apenas como configurar a autenticação.

Configurando Forms Autentication em Asp.Net MVC

Tudo que vou mostrar aqui será na visão MVC, para quem conhece o processo para WebForm verá que o que muda, é que você troca a página do WebForm (aspx) por uma Action ou um Controller, se você deseja que a Action Index seja o foco da autenticação.

Para fazer os exemplos, vou utilizar um novo site padrão do Asp.Net MVC 2, e vou forçar que seja feito o login para acessar a página inicial. O Web.Config para configurar o Forms Autentication fica da seguinte forma:

```
<authentication mode="Forms">
      <forms loginUrl="~/Account/LogOn" timeout="2880" />
</authentication>
```

Nessas linhas, adicionado dentro de system.web, que já estão no projeto padrão, estamos informando no atributo **loginUrl** que a Action responsável por renderizar a View de Login é a **Account/LogOn**, isso quer dizer que sempre que for necessário que o usuário esteja logado para acessar uma determinada Action, o browser será direcionado para esta Action. Também está definido quanto tempo em minutos que o login ficará ativo se o usuário não tiver nenhuma ação no site, no atributo **timeout.**

Mas você vai reparar que mesmo com essas configurações, o site não estará exigindo a **autenticação**. Isso acontece porque você não está definindo as regras de **autorização**. Ou seja, se nada está definido para autorização, qualquer usuário está autorizado a fazer tudo, logo não precisa estar autenticado.

Vamos então definir as regras de autorização, veja que nada muda até agora do que você já conhecia no WebForms, exceto as trocas de páginas por Actions.

Também no system.web, vamos falar que apenas usuário reconhecidos, ou seja, autenticados, podem acessar o site. Para isso, adicionamos as seguintes linhas ao config:

```
<authorization>
     <deny users="?"/>
     <allow users="*"/>
</authorization>
```

Veja que as tags são bem intuitivas. Dentro da tag autorização, eu informo o que eu quero proibir (**deny**) e o que eu quero permitir (**allow**). Por sua vez, dentro dessas duas tags (deny e allow), é comum usarmos dois atributos: **users**, onde coloco os nomes dos usuário que quero permitir ou proibir, e **roles**, onde defini os grupos de usuários que quero permitir ou proibir.

Na configuração que coloquei acima, estou proibindo todos os usuários anônimos (**?**), ou seja, não autenticados, e estou permitindo todos os usuários conhecidos (*), ou seja, autenticados.

Caso não queira ser tão genérico, autorizando todo mundo, você pode usar o nome de um usuário ou grupo, por exemplo:

```
<authorization>
      <deny users="?"/>
      <allow users="fred" roles="admin"/>
</authorization>
```

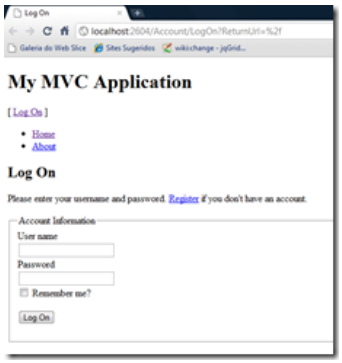
Caso você queira adicionar mais de um grupo ou usuário, é só separar por vírgula:

```
<authorization>
      <deny users="?"/>
      <allow users="fred" roles="admin,users"/>
</authorization>
```

De qualquer forma, após inserirmos as linhas de autorização, nenhuma página será acessível sem login e o site será redirecionado automaticamente para a página configurada na seção **Authentication** do web.config. Porém, se você realizar um teste, verá que ainda não funcionará, pois a tela estará toda torta, como a imagem abaixo:

Category list

- Ajax (7)
- Asp.Net (2)
- C# (2)
- Classes (1)
- Curiosidades (1)
- Design Pattern (2)
- Framework (1)
- Html Control (1)
- JavaScript (8)
- JQGrid (1)
- Jquery (9)
- JSON (1)
- Knockout (3)
- MVC (11)
- PageMethod (1)
- Posts (1)
- Produtos (1)
- ServerControl (2)
- TDD (1)
- Testes (3)
- Validação (1)
- Validator (2)
- Visual Studio (1)
- Web (2)
- WebAPI (1)
- WebForms (5)



Isso acontece porque a segurança do Asp.Net não é apenas para arquivos HTMLs ou Actions, mas para todo o conteúdo do site, incluindo imagens, scripts, css, pastas e sub-pastas, portanto, quando você proíbe o acesso de um usuário ele também não conseguirá acessar as imagens, scripts, etc.

Para que a exibição seja correta, você tem que criar exceções, falando que alguns arquivos ou pastas podem ser acessados mesmo quando o usuário for anônimo, isso é feito adicionando as seguintes linhas ao config,

```
<location path="Content">
    <system.web>
      <authorization>
        <allow users="?"/>
      </authorization>
    </system.web>
  </location>
  <location path="Views/Shared">
    <system.web>
      <authorization>
        <allow users="?"/>
      </authorization>
    </system.web>
  </location>
  <location path="Scripts">
    <system.web>
      <authorization>
        <allow users="?"/>
      </authorization>
    </system.web>
</location>
```

Location são colocadas diretamente abaixo da tag **Configuration** no Web.Config, e servem para configurações específicas para determinadas pastas, arquivos ou Actions. No nosso caso estamos autorizando que usuários anônimos tenham acesso as pastas Content, Shared e Scripts, e se acessarmos o site agora o layout estaria correto:

Assim já temos uma autorização funcionando. Como disse anteriormente, como será realizada a autenticação fica a cargo do desenvolvedor, o Asp.Net já fornece algumas possiblidades, que você pode ver no projeto padrão que vem com o VisualStudio.

Definindo autorização por Action

Acredito que é necessário apenas falar agora de como restringir acesso para cada Action. No MVC isso é muito fácil, podemos utilizar o atributo **Authorize** sobre a Action específica, podendo passar na propriedade **User** ou **Roles** o usuário específico ou grupo específico.

Por exemplo, digamos que eu queira que apenas pessoas do grupo Admin possam criar um novo usuário, poderíamos decorar a Action Create da seguinte forma:

```
1:  // **************************************
2:  // URL: /Account/Register
3:  // **************************************
4:  [Authorize(Roles="admin")]
5:  public ActionResult Register()
6:  {
7:      ViewData["PasswordLength"] = MembershipService.MinPasswordLength;
8:      return View();
9:  }
10:
```

```
11:  [HttpPost,Authorize(Roles="admin")]
12:  public ActionResult Register(RegisterModel model)
13:  {
14:      if (ModelState.IsValid)
15:      {
16:      // Attempt to register the user
17:      MembershipCreateStatus createStatus =
MembershipService.CreateUser(model.UserName, model.Password, model.Email);
18:
19:      if (createStatus == MembershipCreateStatus.Success)
20:      {
21:          FormsService.SignIn(model.UserName, false /* createPersistentCookie
*/);
22:          return RedirectToAction("Index", "Home");
23:      }
24:      else
25:      {
26:          ModelState.AddModelError("",
AccountValidation.ErrorCodeToString(createStatus));
27:      }
28:      }
29:
30:      // If we got this far, something failed, redisplay form
31:      ViewData["PasswordLength"] = MembershipService.MinPasswordLength;
32:      return View(model);
33:  }
```

Perceba que adicionei o atributo **Authorize** as Actions e assim o Asp.Net verificará automaticamente se o usuário logado pertence ao grupo informado.

É importante ressaltar que tudo fica automático porque estamos utilizando o **Membership Provider** do Asp.Net, que já sabe como fazer para verificar se um usuário faz parte ou não de um grupo. Caso você não utilize o **AspNetSqlMembershipProvider**, isso não vai ficar tão automático assim, pois você terá que criar um **Membership Provider**, que também não é nada complicado, e é assunto para um próximo post.

Utilizei o **AspNetSqlMembershipProvider** porque quando criamos um novo projeto em Asp.Net MVC ele automaticamente utiliza esse provider, é muito fácil de utilizar e eu aconselho a utilização dele em qualquer projeto que não tenham regras muito específicas de segurança..

Bom pessoal, é isso! Acredito que com isso você já consiga fazer um site completo em MVC sem muita dificuldade.

Qualquer dúvida é só postar ai nos comentários, não vou postar um arquivo para download porque é exatamente o projeto padrão, com as alterações mostradas nos exemplos.

Até o próximo!

Tags: Autenticação, Autorização

Categorias: MVC

5.0 ponto(s). Avaliado por 1 pessoas

31. março 2011 07:03 by Frederico B. Emídio | Comentários (4) | Permalink

Posts relacionados

Executando serviço em Background na infraestrutura do Asp.Net
Olá pessoal! Hoje eu vou abordar um tema não muito ortodoxo: As possíveis formas de “hospedar” um s...

HTML + JavaScript + CSS
Olá pessoal! Continuando nossa série de posts introdutório sobre Asp.Net MVC, vamos falar hoje das ...

MVVM e Asp.Net MVC com KnockoutJS
Introdução Hoje vou falar de um tema novo, porém, muito importante para quem desenvolve sistemas We...

Comentários

Junior luiz

Frederico estou estudando mvc eu tenho um banco com tabelas onde faço a autenticação em w ebforms, se eu quiser fazer o mesmo em mvc e tendo a facilidade do [authorize] isso é possível e como eu faria ?

15/04/2011 17:40:38 #

Frederico

Sim, é possível, publicarei em breve um artigo mostrando todo o processo de autenticação e autorização em uma infraestrutura totalmente particular, onde o próprio desenvolvedor é responsável por criar a estrutura onde armazenará os usuários e grupos.

Aguarde a publicação em breve.
Abs,

Frederico

16/04/2011 02:01:38 #

Marcos Batista

Frederico!
Então, quando eu autentico o usuário (Verifico o loguin e a senha dele se está correto com linq), eu atribuo um objeto do tipo usuário com Membership Provider, tipo no caso de sessão? Quando eu necessitar de alguma parte desse objeto é só instânciar? Aguardo! Obg

08/12/2012 19:14:17 #

Frederico

Olá Marcos.

Se você atribuir esse objeto para a sessão (foi o que eu entendi que você faz) é só recuperá-lo, sem precisar criar uma nova instância.

Se essa não é a sua dúvida, comente de novo.

Abs
Frederico

12/12/2012 05:04:21 #

Comentar

Nome*

E-mail*

Site

Comentário | Pré-visualização

b i u quote

Me avise quando alguém comentar este post

Salvar comentário