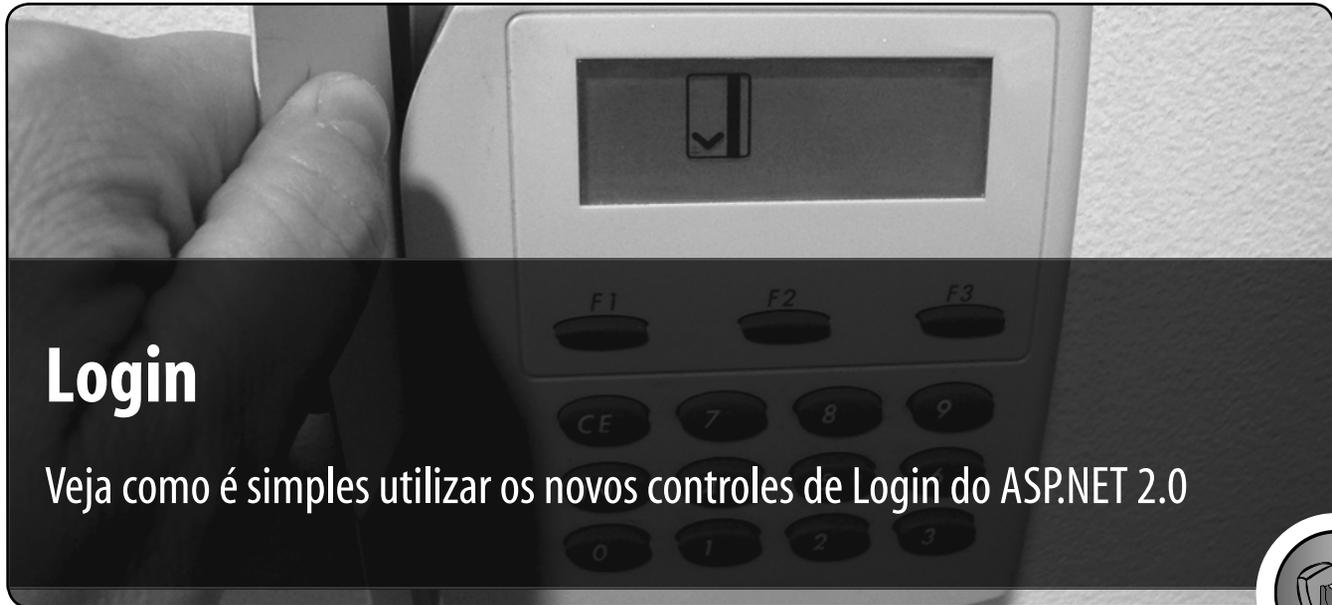


## Nesta seção você encontra artigos intermediários sobre Delphi Win32 e Delphi for .NET



### Login

Veja como é simples utilizar os novos controles de Login do ASP.NET 2.0



#### Guinther Pauli

([guinther@devmedia.com.br](mailto:guinther@devmedia.com.br))

é Microsoft Certified: MCP, MCAD, MCS.D.NET, MCTS, MCPD, Borland Certified: Delphi 6, 7, 2005, 2006, Web e Kylix. Editor Geral das Revistas .net Magazine e ClubeDelphi. Desenvolve em ASP.NET há 8 anos.



#### Adriano Santos

([falecom@adrianosantos.pro.br](mailto:falecom@adrianosantos.pro.br))

é desenvolvedor Delphi desde 1998. Professor e programador PHP. Bacharel em Comunicação Social pela Universidade Cruzeiro do Sul, SP. É Editor Técnico, Colunista e Membro da Comissão Editorial das revistas ClubeDelphi. Gerente de Desenvolvimento da SoftPark, empresa parceira Borland. Mantém o blog Delphi to Delphi ([www.delphitodelphi.blogspot.com](http://www.delphitodelphi.blogspot.com)) com dicas, informações e tudo sobre desenvolvimento Delphi

Nas últimas edições da revista ClubeDelphi vimos alguns artigos voltados à segurança. Vimos artigos mostrando como lidar com usuários no *Firebird* e aplicações Win32 e até dicas de segurança no PHP. O que veremos nesse artigo tem tudo a ver com segurança. O *.NET Framework 2.0* está repleto de funcionalidades que visam agilizar o trabalho do desenvolvedor, incluindo controles de *login*. Um exemplo claro disso são os controles de *Login* do *ASP.NET 2.0*.

Aliando o uso dos controles de *Login* com o novo esquema de segurança do *ASP.NET 2.0*, podemos implementar segurança em nossas aplicações sem uma linha de código sequer, como veremos neste artigo. Isso mesmo! Com o *ASP.NET 2.0* podemos criar telas para incluir usuários, solicitar *login* ao acessar determinada página, modificar senha etc. O mais interessante disso é que não precisaremos programar praticamente nada em nossa aplicação.

#### Resumo DevMan

Os novos controles de Login do ASP.NET 2.0 facilitam e muito a realização de tarefas rotineiras no que diz respeito à segurança de um Web Site: criação de usuários de grupos, regras de acesso, troca de senha, lembrar senha, visualização de parte do site de acordo com o grupo e muito mais.

#### Nesse artigo veremos

- Utilização dos controles de Login;
- Integração do *aspnetdb* com o banco de dados da aplicação.

#### Qual a finalidade

- Demonstrar a facilidade no uso de controles de login do ASP.NET 2.0.

#### Quais situações utilizam esses recursos?

- Em praticamente todos os sites que necessitem de autenticação.

#### Links

Neste artigo, utilizaremos o SQL Server 2005 Express, free, comprovadamente o MELHOR banco para se trabalhar com aplicações ASP.NET.

#### SQL Server 2005 Express

<http://www.microsoft.com/express/download/>

## Controle de Login

Para iniciarmos nosso estudo, vamos criar uma pequena aplicação capaz de demonstrar todos os recursos de *login* no ASP.NET 2.0 utilizando o RAD Studio 2007. Crie uma nova aplicação usando o menu *File>New>ASP.NET Web Application – Delphi .NET*. Em *Name* digite *Login*, que será o nome de nossa aplicação, e em *Location* indique o local onde os fontes do sistema serão salvos. Mantenha o padrão (*C:\inetpub\wwwroot\*), pois o diretório *\inetpub\wwwroot* do Windows já possui todas as informações de acesso e segurança configuradas no IIS (Internet Information Server). Troque apenas o nome da pasta do projeto se preferir. Mantenha o servidor de testes como IIS e confirme (Figura 1).

A primeira coisa que vamos observar é a paleta de componentes *Web Login* (Figura 2). Como podemos ver, temos uma série de controles específicos para criação de *login* em nossos sistemas Web.

Arraste o controle *Login* para sua página *Default.aspx*. Veja que temos todos os elementos de um componente que efetua *logins* de um site. Como qualquer controle ASP.NET, podemos alterar o formato de *layout* do controle acessando a opção *Auto Format* que encontramos em *Tasks*. Em nosso exemplo foi aplicado o formato *Professional*. Você pode conferir o resultado na Figura 3.

Adicione um novo *Web Form* ao projeto chamado *WebMenu.aspx*. Essa será a página que chamaremos caso o usuário informe um usuário e senha válidos. Para isso acesse o menu *File>New>Other>Delphi for .NET Projects>New ASP.NET Files>ASP.NET Page* e *design* do *Menu.aspx*, inclua o texto *Olá, você está logado!*. Antes da vírgula inclua o controle *LoginName*.

Volte ao *Default.aspx*, clique no *Login* e vá em propriedades. Procure a propriedade *DestinationPageUrl*, clique no botão com três pontos e escolha a página *WebMenu.aspx*. Execute seu projeto, informe em *UserName* o usuário da sua máquina local e sua respectiva senha. Veja o resultado na Figura 4.

Esse erro ocorre porque ainda não configuramos nada no ASP.NET Configuration Site, que já veremos a seguir. Antes de mais nada, é possível traduzirmos

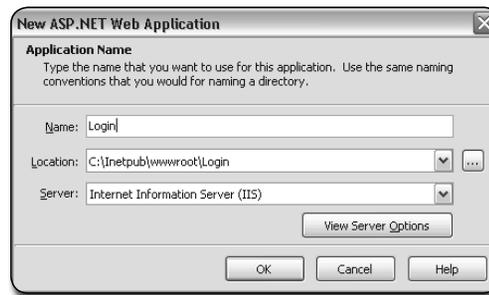


Figura 1. Criação do projeto de login

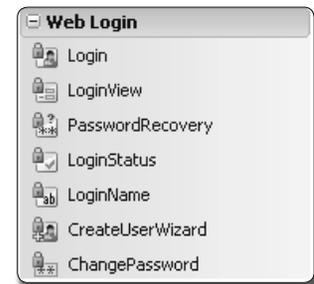


Figura 2. Controles de Login do ASP.NET 2.0

as mensagens que aparecem nas telas de *login*. Basta acessar a propriedade *FailureText* do componente e efetuar a mudança. Para traduzir os rótulos das caixas de texto, caixas de opções e botões acesse as propriedades: *UserNameLabelText*, *PasswordLabelText*, *RememberMeText* e *LoginButtonText*, respectivamente.

## ASP.NET Configuration

Através do ASP.NET Configuration Site podemos trabalhar toda a segurança da aplicação. Temos condições de criar usuários, adicionar permissões ou negar acesso a determinadas páginas, incluir grupos de regras (*roles*) e uma série de outros recursos de segurança.

Vamos entender agora como funciona o novo esquema de segurança do ASP.NET 2.0. Vá até o *Project Manager*, clique com o botão direito e escolha a opção *ASP.NET Configuration Site*. O browser será aberto e nele uma tela como a Figura 5 será exibida.

Clique na guia *Security*, onde a primeira configuração necessária é a autenticação. Utilizaremos o *wizard* para nos auxiliar no processo de configuração. Clique em *Use the security Setup Wizard to configure security step by step*. Clique em *Next*, pois a primeira tela que visualizamos é apenas um resumo das tarefas que serão realizadas durante o assistente. Logo em seguida configuraremos o modo como nossa aplicação fará a autenticação. Temos duas opções:

- *Internet (Forms)*: a autenticação é baseada em um formulário de *login* e a autenticação feita “na mão” pelo desenvolvedor;

- *Intranet (Windows)*: a autenticação é baseada no *login* na rede, ou seja, não necessita de nenhum trabalho de auten-



Figura 3. Controle Login na Default.aspx

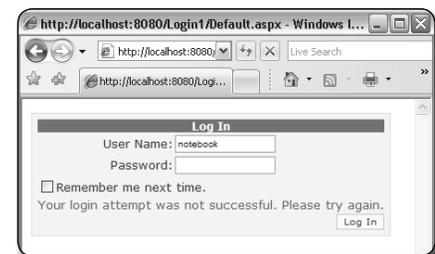


Figura 4. Falha ao tentar logar no site

### Nota

Quando criamos uma aplicação Web no RAD Studio 2007, ele nos cria o arquivo *Web.Config* totalmente pré-configurado para acessar o banco de dados BlackFish da própria CodeGear. Portanto, ao iniciar uma nova aplicação no RAD teremos que modificar esse arquivo para que não ocorram erros inesperados ao executar o Web Site. Vejamos como fazer isso.

Acesse o arquivo *Web.Config* no *Project Manager* e localize, primeiramente, a *tag <connectionStrings>*. Remova todo o conteúdo dela deixando-a apenas como mostrado a seguir:

```
<connectionStrings>
</connectionStrings>
```

Logo abaixo de *<connectionStrings>* encontraremos outra *tag* que precisa ser removida, *<system.data>*. Limpe-a também deixe-a como a seguir:

```
<system.data>
</system.data>
```

Em seguida, ao final do arquivo, localize as *tags <profile>* e *<roleManager>*. No conteúdo dessas *tags* você encontrará um atributo chamado *connectionStringName*. Troque seu nome para *LoginDevMedia* em ambas.

Pronto! Com isso podemos rodar normalmente nossa aplicação.

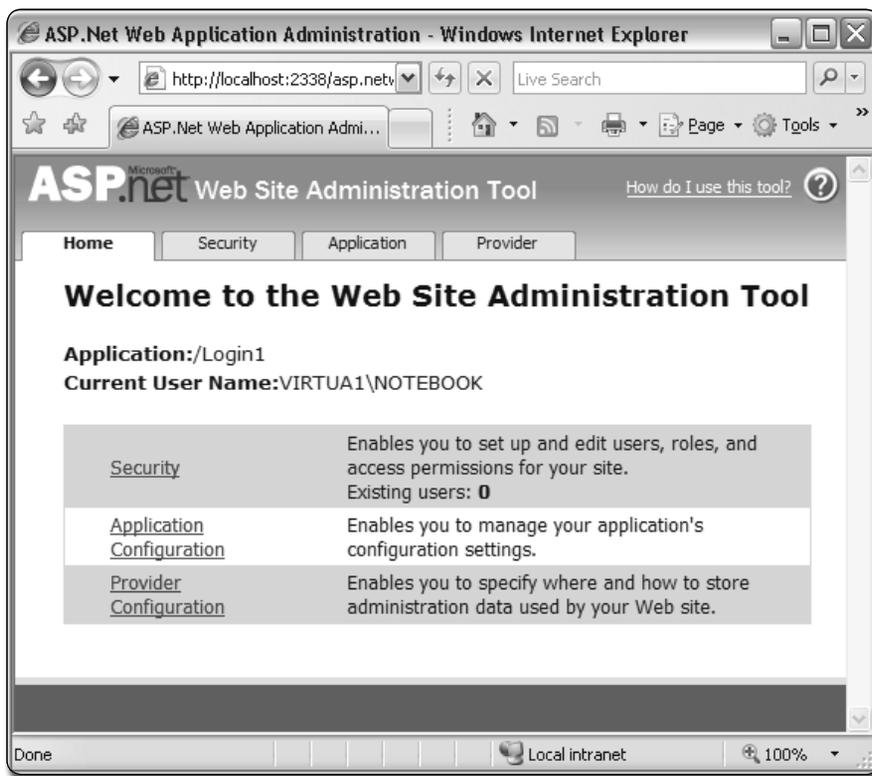


Figura 5. Página inicial do ASP.NET Configuration Site



## Nota do DevMan

Por padrão os componentes da paleta *WebLogin* possuem seus *Label's* em inglês. Isso pode ser facilmente modificado, pois há diversas propriedades que podemos modificar e traduzir para a linguagem local. Vejamos o que podemos modificar para que o usuário não se perca na hora de efetuar o *login* ou mesmo cadastrar-se. Pressione *F11* com o componente *LoginView* selecionado e veja suas propriedades:

- *UserNameLabelText*: Troque de *User Name*: para "Nome de Usuário:";
- *PasswordLabelText* Modifique para "Senha:";
- *PasswordRequiredErrorMessage*: "Senha obrigatória";
- *UserRequiredErrorMessage*: "Nome de Usuário obrigatório";
- *RememberMeText*: "Lembrar meu usuário";

Note que, ao modificar a propriedade, o *template* é automaticamente atualizado na *IDE* do *RAD Studio 2007*. Além disso, perceba que outras propriedades podem sofrer modificações, tais como: *FailureText*, texto apresentado quando alguma falha de *login* ocorre e *LoginButtonText*, ou seja, o texto do do botão *Login*.

A seguir veremos a inclusão do controle *CreateUserWizard* que também pode ser altamente modificado pra a linguagem local. Você contará com algumas propriedades importantes, como por exemplo:

- *UserNameLabelText*: Texto para o *Label* do *TextBox* de nome de usuário;
- *UserNameRequiredErrorMessage*: Mensagem apresentada ao requerer nome de usuário;
- *UnknownErrorMessage*: Mensagem de erro desconhecido;
- *QuestionLabelText:Label* para o campo de pergunta secreta;
- *QuestionRequiredErrorMesasge*: Mensagem apresentada ao requerer nome de usuário;
- *PasswordLabelText*: Texto para o *Label* do *TextBox* de senha;
- *PasswordRequiredErrorMessage*: Mensagem de texto ao requerer senha;

E ainda o textos dos botões:

- *ContinueButtonText*: Texto do botão *Continue* ("Continuar");
- *CreateButtonText*: Texto do botão *Create User* ("Confirmar criação de usuário");
- *FinishButtonText*: Texto do botão *Finish* ("Finalizar");
- *StartNextButtonText*: Texto do botão *Next* ("Próximo passo");

Entre outras mensagens e textos que podem ser modificados. Em resumo, os controles do *ASP.NET 2.0* podem ser personalizados em praticamente tudo, facilitando a vida do desenvolvedor e agilizando o processo de criação e manutenção do site.

ticação. É mais utilizado em aplicações corporativas.

Vale lembrar que muitas configurações de segurança usarão um *BD* especial criado automaticamente pelo *ASP.NET*, o *aspnetdb*. Novamente clique em *Next*. Faremos agora algumas configurações avançadas. Habilite a opção *Enable roles for this Web Site*, dessa forma estamos prevendo que nossa aplicação será capaz de validar grupos de usuários para ter acesso a determinadas partes do site.

**Nota:** Não precisamos alterar nada em relação ao armazenamento, vamos utilizar o padrão do *ASP.NET*. O *ASP.NET* cria automaticamente um banco no *SQL Server Express*, chamado *aspnetdb*, que armazena as informações de configuração das aplicações criadas, inclusive todas as informações relacionadas à segurança.

Prosseguindo com o *wizard*, nessa etapa temos que definir se nosso *Web Site* trabalhará com *Roles*, como já foi mencionado. Clicando novamente em *Next*, o assistente solicitará o nome do grupo para que ele possa criar para nós. Os usuários são agrupados em *Roles* para identificar que eles desempenham as mesmas funções na empresa, como por exemplo: *Vendas*, *Marketing*, *Administradores* etc. Use *Roles* em aplicações com muitos usuários, pois você pode aplicar regras de segurança diretamente a esses grupos e então incluir usuários a elas. Isso facilita a manutenção dos usuários da aplicação. Para criar uma *Role* basta informar o seu nome no campo *New Role Name* e clicar no botão *Add Role*. Para o nosso exemplo vamos criar algumas *Roles*: *Vendas*, *Marketing*, *Administradores* e *Usuarios*. Veja na **Figura 6** que as *Roles* criadas aparecem em uma lista, onde podemos inclusive excluí-las, através do link *Delete*. Após criar as *Roles* clique em *Next*.

## Criando usuários

Agora temos a opção de criar usuários. Vamos criar alguns usuários para podermos realizar nossos testes. Podemos apenas informar o nome do usuário, senha, e-mail e adicionalmente uma pergunta e resposta para auxiliar a lembrar da senha em caso de esquecimento. Ao

terminar o preenchimento dos campos clique em *Create User*. Você pode criar dois usuários com os nomes e demais informações da sua escolha (Figura 7). Após criar os dois usuários necessários para nosso exemplo, clique em *Next*.

**Nota:** Por padrão, o ASP.NET usa uma política de segurança para as senhas, que devem ser maiores que sete dígitos e requer um caractere especial (\*, &, % etc.).

### Access Rules

Nessa última etapa do *wizard* podemos definir as *Access Rules*. Não confunda *Role* com *Rule*. As *Access Rules* são regras de acesso, que podem servir para permitir ou negar o acesso a determinadas áreas do site. Você pode utilizar as *Access Rules* para controlar o acesso de toda a aplicação ou de cada um das *Folders* (pastas) da aplicação. As regras de acesso podem ser aplicadas a um único usuário, a uma *Role*, a todos os usuários ou somente a usuários anônimos.

Para adicionar uma *Access Rule* selecione primeiramente a pasta na qual ela será aplicada. Se a *Access Rule* for válida para toda a aplicação, selecione o *Folder* raiz. Em seguida defina se a *Access Rule* será aplicada a uma *Role*, usuário, todos os usuários ou somente os usuários anônimos. Para finalizar você precisa definir se a *Access Rule* irá garantir o acesso à pasta ou negá-lo. Após definir esses três pontos, clique em *Add This Rule*.

As *Rules* adicionadas irão aparecer em uma lista logo abaixo. As regras de acesso são aplicadas na ordem dessa lista, e vale a primeira que for encontrada, por exemplo: se a primeira *Rule* nega o acesso aos usuários anônimos na aplicação, e a segunda garante, fica valendo a primeira. Nessa lista podemos excluir as *Rules* através do link *Delete*.

Em nosso exemplo iremos apenas criar duas *Access Rules* simples. Uma garantirá acesso aos usuários da *Role Vendas*, e a outra que irá garantir o acesso aos usuários da *Role Marketing* (Figura 8).

Após a criação das *Access Rules*, clique em *Next*. Uma tela de conclusão irá aparecer, basta agora clicar em *Finish*. Você pode gerenciar usuários, *Roles* e *Access*

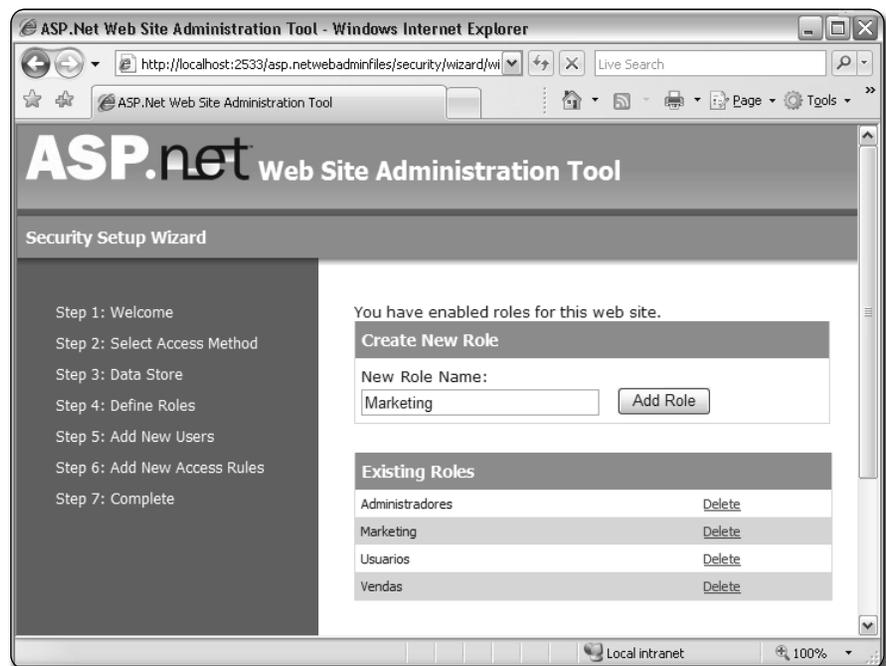


Figura 6. Criando Roles

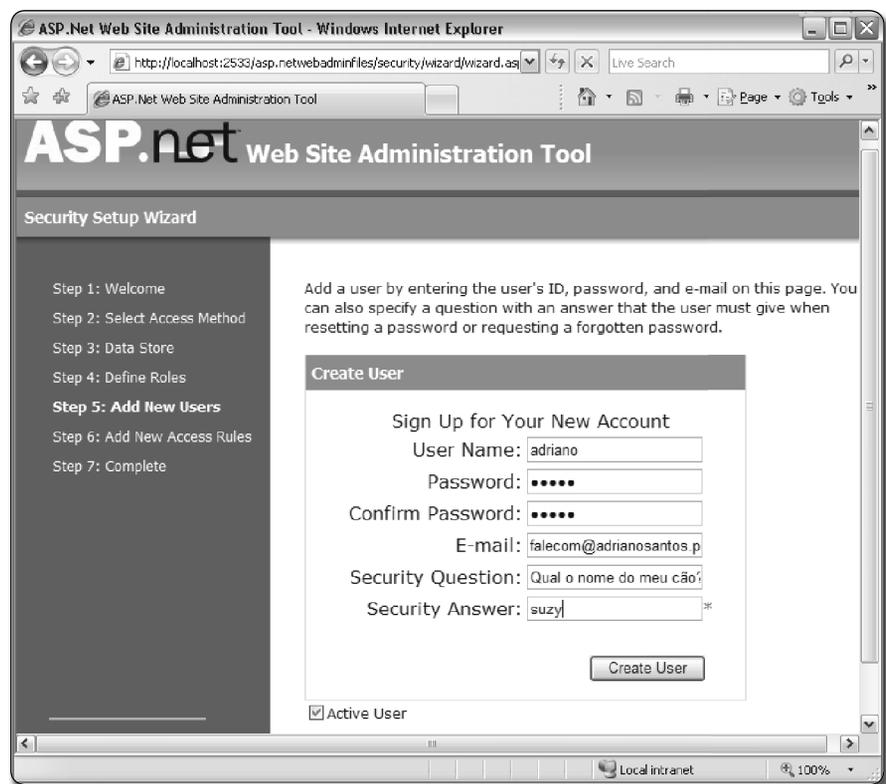


Figura 7. Criando um novo usuário no ASP.NET 2.0



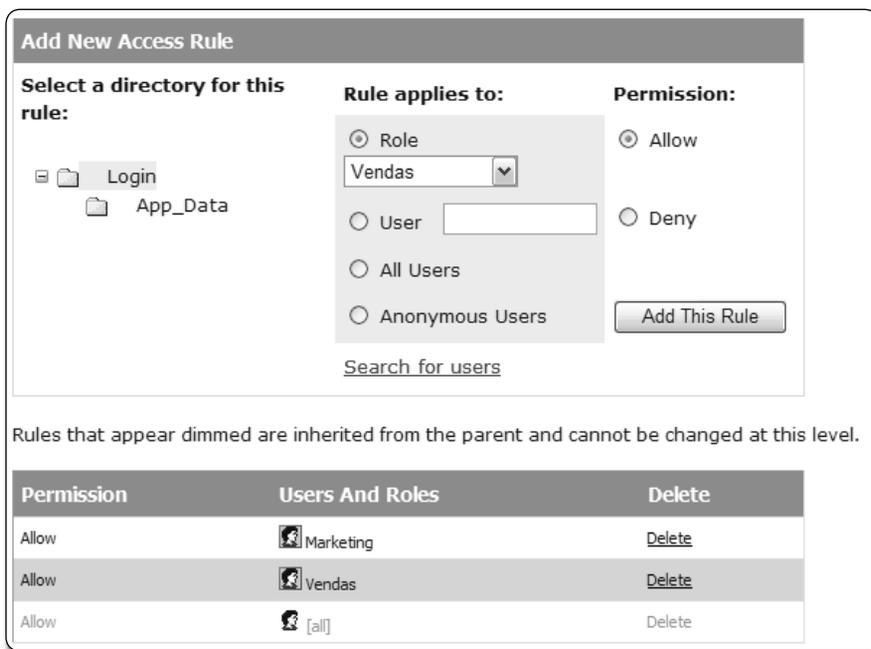


Figura 8. Definindo Access Rules

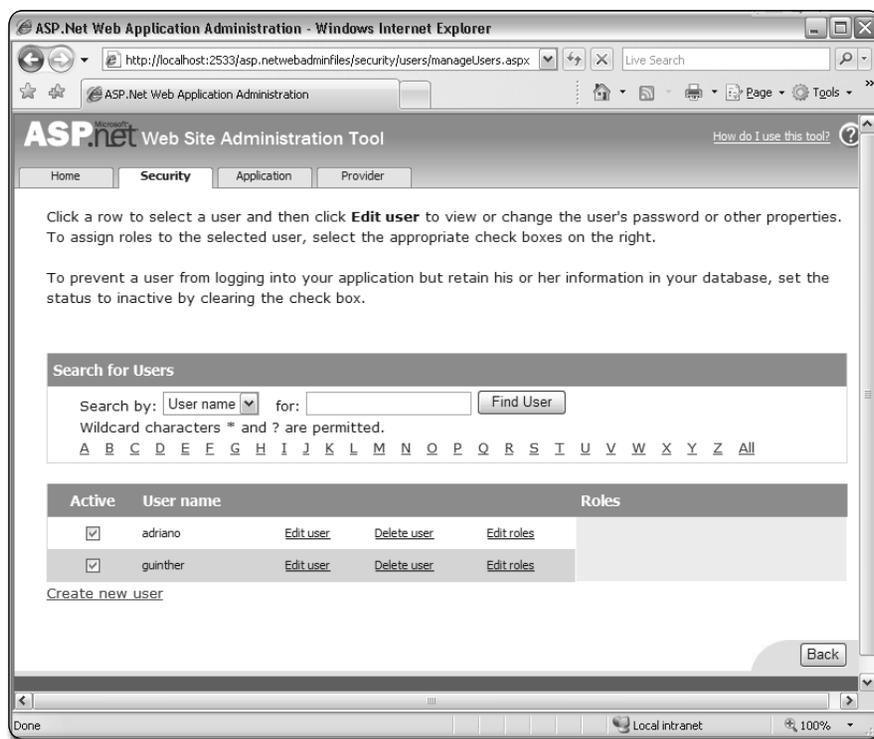


Figura 9. Definindo Roles dos usuários

Rules no momento que quiser, basta acessar o *ASP.NET Configuration* e ir até a aba *Security*

### Atribuindo usuários às roles

Você deve ter percebido que está faltando algo em nossa configuração. Ainda não definimos qual é a *Role* de cada um dos usuários que criamos. É exatamente isso que faremos agora. Na aba *Security*, do *ASP.NET Configuration*, clique na opção *Manage Users*. Uma tela como a da **Figura 9** aparecerá.

Para definir quais são as *Role* de um usuário, você deve clicar no link *Edit roles*, que fica ao lado de cada usuário da lista de usuários. Ao clicar no link, à direita nos aparece uma lista com todas as *Roles* disponíveis. Basta selecionar as *Roles* do usuário em questão. Em nosso exemplo, vamos definir que o usuário *adriano* participa da *Role Vendas*, e que o usuário *gunther* participa da *Role Marketing*

### Testando o acesso do usuário

Com certeza, agora nosso sistema já funcionará perfeitamente após as configurações no *ASP.NET Configuration Site*, porém para termos certeza disso vamos fazer um pequeno teste. Volte ao *RAD Studio 2007*, compile e execute a aplicação. No controle de *Login*, informe o usuário *adriano* ou *gunther* e sua respectiva senha. Veja o resultado na **Figura 10**.

### Controle LoginView

O controle *LoginView* é um dos controles mais interessantes do *ASP.NET 2.0*. Com ele podemos determinar quais áreas cada grupo de usuários tem acesso e conseqüentemente quais controles de tela aparecerão. Vamos então utilizar o controle *LoginView* para validarmos a utilização das *Roles*. O *LoginView* é um controle utilizado para definir diferentes conteúdos em determinadas áreas de uma página, de acordo com a *Role* do usuário, assim como já mencionado. Abra a página *WebMenu.aspx*, e inclua um controle *LoginView*. Nas *Tasks* do controle clique na opção *Edit RoleGroups*. Aparecerá uma janela como demonstrada na **Figura 11**.



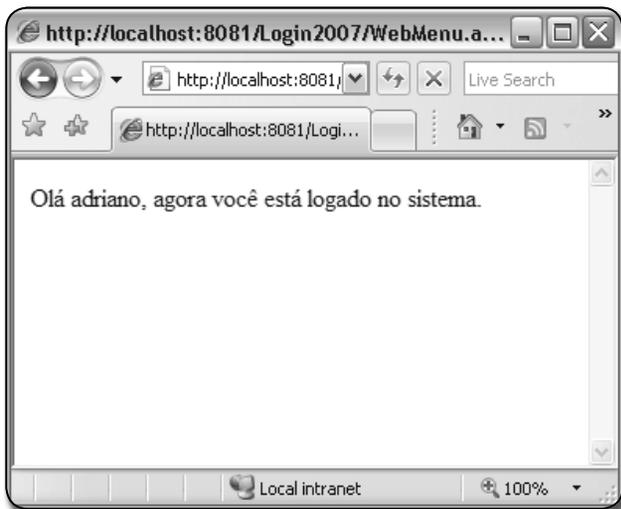


Figura 10. Usuário autenticado na aplicação

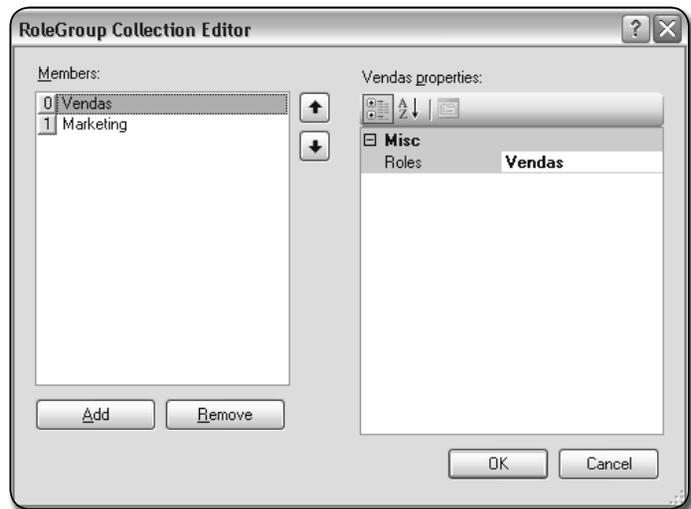


Figura 11. Configurando RoleGroups no LoginView

Clique em *Add* e aparecerá um item chamado *RoleGroup*. À direita teremos uma propriedade chamada *Roles*, clique no botão com três pontos referente a essa propriedade. Aparecerá uma janela para informarmos o nome da *Role*. Digite *Vendas* e clique em *OK*. Clique novamente em *Add* e repita esse procedimento para a *Role Marketing*. Ao finalizar clique em *OK*.

Ainda em *Tasks*, observe a opção *Views*. Como mostra a **Figura 12**, podemos escolher diversas *Views* possíveis e definir um conteúdo diferente para cada uma delas.

Vamos selecionar a *View* de *Vendas*. Veja que um *LoginView* funciona como um *container*, onde podemos inserir controles dentro dele. Vamos apenas inserir o texto *Esse é o conteúdo para quem é de Vendas*. Modifique para a *View* do *Marketing* e insira o texto *Esse é o conteúdo para quem é de Marketing*.

Feitas essas alterações, salve, compile e execute. Acesse primeiramente, utilizando o usuário *adriano*. Veja que ele tem acesso apenas ao conteúdo da *Role*

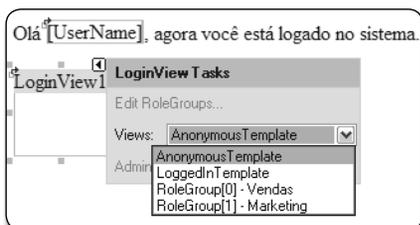


Figura 12. Possíveis Views do controle LoginView

# PENSE...

QUANTO TEMPO  
VOCÊ GASTARIA  
PARA DESENVOLVER  
COBRANÇA COM BOLETOS  
BANCÁRIOS PARA  
APENAS UM BANCO  
NO SEU SOFTWARE

# COBREBEMX

-  56 BANCOS E MAIS DE 430 CARTEIRAS DE COBRANÇA PARA IMPRESSÃO E/OU ENVIO DE BOLETO BANCÁRIO POR EMAIL;
-  GERAÇÃO DE BOLETOS ON LINE;
-  GERAÇÃO E LEITURA DE ARQUIVOS (REMESSA/RETORNO) NOS PADRÕES FEBRABAN E CNAB;
-  MAIS DE 40 EXEMPLOS EM DIVERSAS LINGUAGENS DE PROGRAMAÇÃO



DOWNLOADS E INFORMAÇÕES EM [WWW.COBREBEM.COM](http://WWW.COBREBEM.COM)

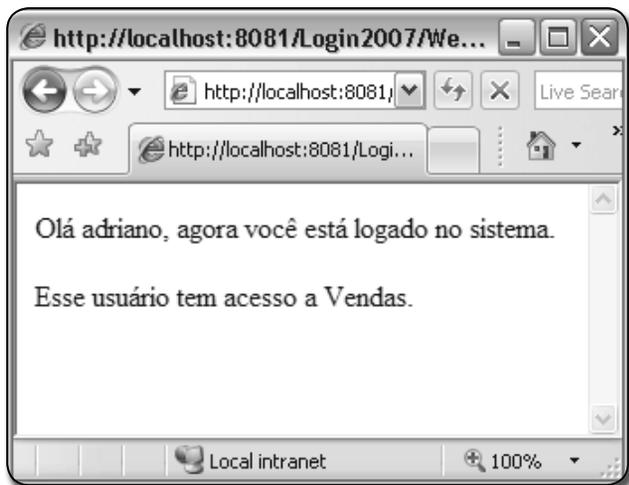


Figura 13. Usuário do grupo Vendas

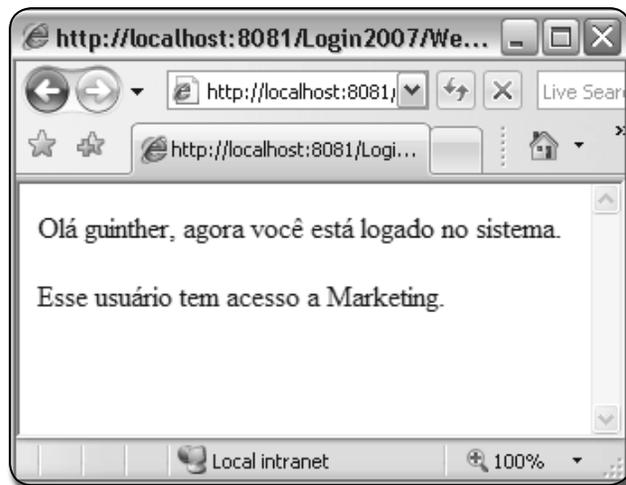


Figura 14. Usuário do grupo Marketing

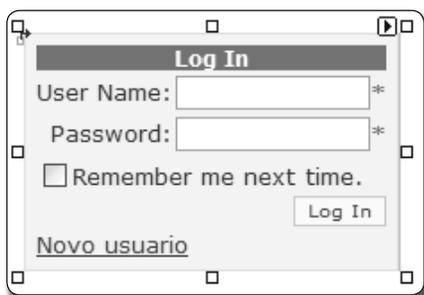


Figura 15. Controle de Login com opção de cadastrar novo usuário

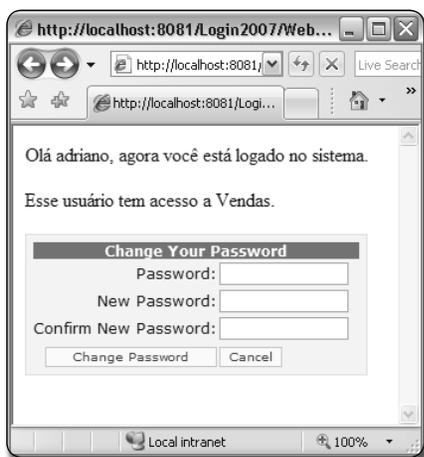
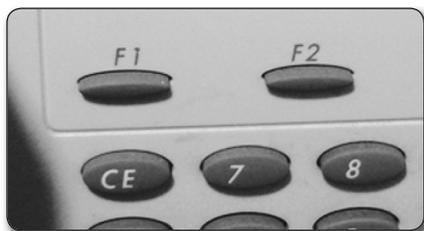


Figura 16. Modificando a senha com o controle ChangePassword



de *Vendas*. Saia e acesse com o usuário *guinther*. Veja que ele apenas acessa o conteúdo da *Role* de *Marketing*. É possível também definir um conteúdo diferente para usuários anônimos, ou simplesmente para usuários *logados* que não pertençam a nenhuma *Role* configurada (Figuras 13 e 14).

### Controle CreateUserWizard

Vamos utilizar agora um controle *CreateUserWizard*. Adicione um novo *WebForm* ao projeto chamado *CreateUser.aspx*. No *design*, insira um *CreateUserWizard*. Configure o *AutoFormat* para *Professional*. Nas propriedades do controle e na propriedade *ContinueDestinationPageUrl* informe a página *Default.aspx*, para indicar a página de destino após a criação da nova conta.

Volte na página *Default.aspx* e vá até as propriedades do controle de *Login*. Localize a propriedade *CreateUserText* e digite *Novo Usuário*. Logo abaixo, na propriedade *CreateUserUrl*, informe a página que acabamos de criar, usando o botão de reticências. Veja na Figura 15 como fica nosso formulário de *login*. Salve, compile e execute o projeto. Perceba que ao clicar em *Novo usuário* o formulário para criação de usuário é chamado e podemos nos cadastrar normalmente no site agora.

Clique no link *Novo usuário* e veja que podemos incluir um novo usuário, assim como fizemos pelo *ASP.NET Configuration*. Preencha o formulário para

a criação do novo usuário e clique em *Create User*. Irá aparecer uma mensagem informando que o usuário foi criado com sucesso, clique no botão *Continue*. Você voltará à tela de *login*, informe o usuário e senha que você acabou de criar e clique em *Login*. Pronto, você já tem um novo usuário habilitado!

### Controle ChangePassword

Temos ainda a possibilidade de permitir que a senha seja alterada usando o controle *ChangePassword*. Com esse controle o usuário modifica sua própria senha sem a necessidade de entrar em contato com o administrador do site. Vamos ao exemplo: apenas acione o *design* da página *WebMenu.aspx*. Inclua um controle do tipo *ChangePassword* e configure o seu formato para *Professional* clicando no link *Auto Format*. Salve, compile e execute. Informe o usuário e senha no controle de *Login*. Veja que em nossa página *WebMenu.aspx* já temos disponível um controle para modificação de senhas (Figura 16). Faça um teste e modifique a senha do usuário.

### Integrando a segurança do ASP.NET em seu banco

Certamente você deve estar se perguntando como integrar esse esquema de segurança em seu próprio banco de dados, uma vez que o *ASP.NET* armazena seus usuários no *aspnetdb*. Nos exemplos que fizemos até agora, utilizamos o *aspnetdb*, que foi automaticamente criado no *SQL*

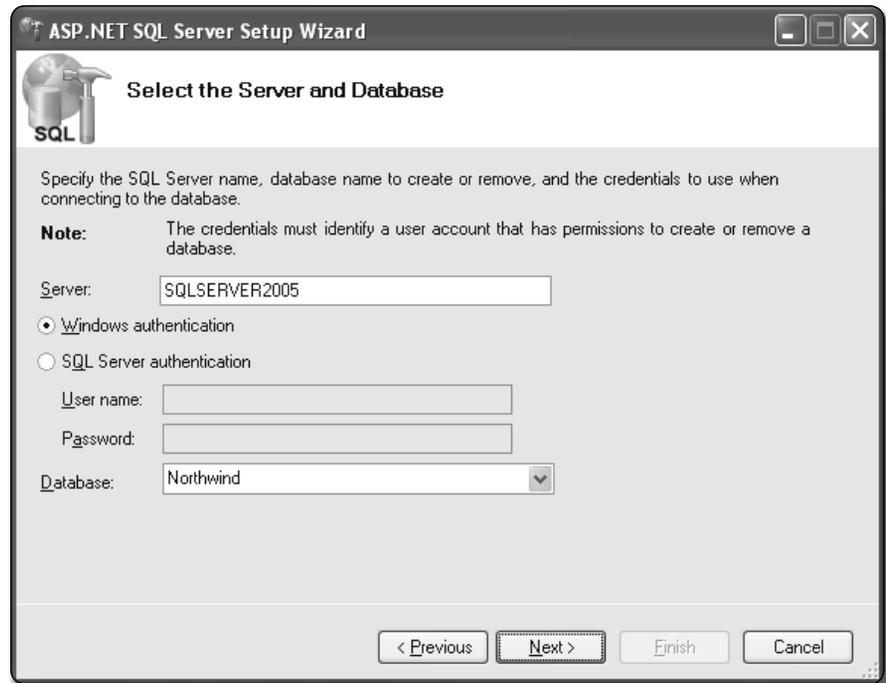
Server Express. Esse é um tipo de solução possível, mas também é possível integrar todo o esquema de segurança do ASP.NET em nosso próprio banco.

Quando o ASP.NET 2.0 é instalado, temos um aplicativo chamado *aspnet\_regsql.exe*, que fica no diretório *C:\Windows\Microsoft.NET\Framework\v2.0.50727*. Através desse aplicativo podemos criar toda a infra-estrutura de tabelas de segurança em nosso próprio banco do SQL Server. Essa estrutura de que falamos é a que foi demonstrada anteriormente na **Figura 7**.

Vamos fazer um teste criando essa estrutura em um banco diferente, nesse caso em nosso próprio banco.

**Nota :** O download, instalação e criação do banco de dados não serão abordados aqui, pois fogem do escopo desse artigo.

Pelo *prompt* de comando vá até o diretório indicado e execute o aplicativo



**Figura 17.** Definindo o banco Northwind para criação da estrutura do Application Services

**A EDIÇÃO QUE VOCÊ PRECISA ESTÁ ESGOTADA?**

**SEUS PROBLEMAS ACABARAM!!**

**Seja um assinante Gold!**

Com a assinatura Gold você já pode consultar online todos os artigos publicados na sua revista desde a edição nº 1.

DevMedia GROUP

Saiba Mais! Acesse: [www.devmedia.com.br/assgold](http://www.devmedia.com.br/assgold)  
Para mais informações: [www.devmedia.com.br/central](http://www.devmedia.com.br/central)

Assinatura **Gold**

Atenção: Já encontram-se disponíveis as assinaturas GOLD das revistas WebMobile e .net Magazine.

*aspnet\_regsql.exe*. Um *wizard* será iniciado. Clique em *Next* para prosseguir. Na próxima tela o *wizard* pergunta se queremos criar a estrutura do *Application Services* ou se queremos removê-la. Vamos selecionar a primeira opção e clicar em *Next*.

**Nota:** É importante salientar que a estrutura de tabelas criada pelo *ASP.NET* não se destina apenas à segurança, e sim a um conjunto de serviços (*Application Services*) que podemos implementar em nossas aplicações.

Na próxima janela devemos definir o servidor e o banco *SQL Server* onde criaremos a estrutura, como mostra a **Figura 17**. Veja que definimos o banco *Northwind*, um banco padrão do *SQL Server* usado para testes, semelhante ao *Employee* do *Firebird*. Em seguida clique em *Next*. A próxima janela é apenas uma

confirmação, clique em *Next* novamente e aguarde alguns instantes enquanto a estrutura é criada. Em seguida, basta clicar em *Finish*. As tabelas do *ASP.NET* foram agora criadas no banco *Northwind*.

O *ASP.NET* acessa e persiste as informações do *Application Services* através de classes chamadas de *Providers*. Precisamos agora configurar nossa aplicação para que o *provider* responsável pelo acesso aos dados de segurança da nossa aplicação acesse o banco *Northwind* e não o *aspnetdb*. Você deve configurar o seu arquivo *web.config* como mostra a **Listagem 1**.

Observe que temos uma *ConnectionString* que aponta para o *Northwind*, e logo abaixo temos a configuração do *provider* *AspNetSqlMembershipProvider* e do *provider* *AspNetSqlRoleProvider*. Veja que estamos apontando para a *ConnectionString* criada, e que também podemos definir uma série de confi-

gurações de segurança aqui, como por exemplo, o tamanho mínimo das senhas, no atributo *minRequiredPasswordLength*. Pronto, nossa aplicação e banco já têm a segurança do *ASP.NET* implementada! Basta agora criar os usuários, *Roles* e *Access Rules* necessários.

Caso queira, também pode criar relacionamentos entre as tabelas do nosso sistema com a tabela *aspnet\_Membership*. Dessa forma, como exemplo, podemos identificar que o usuário logado é determinado funcionário da tabela *Employees* ou determinado cliente da tabela *Customers*, e aí aplicar as regras de negócio da nossa aplicação. Tudo depende da necessidade e realidade de cada solução.

**Nota:** Adicionalmente, é possível customizar os *providers* de *Membership* e *Roles*, para que acessem uma estrutura de tabelas diferente da criada e utilizada pelo *ASP.NET*. Isso é útil caso nosso banco não seja um *SQL Server*, ou caso queiramos aproveitar uma tabela de usuários já existente.

**Listagem 1.** Web.config alterado para configurar Provider e ConnectionString

```
<?xml version="1.0" encoding="utf-8"?>
<configuration xmlns="http://schemas.microsoft.com/.NetConfiguration/v2.0">
  <connectionStrings>
    <clear/>
    <add
      name="SqlConnectionString"
      connectionString="Data Source=SQLSERVER2005;Initial Catalog=NorthWind;Integrated Security=True;"/>
    </connectionStrings>

  <system.web>
    <authentication mode="Forms" />
    <membership defaultProvider="AspNetSqlMembershipProvider">
      <providers>
    <clear/>

    <add
      name="AspNetSqlMembershipProvider"
      type="System.Web.Security.SqlMembershipProvider"
      connectionStringName="SqlConnectionString"
      applicationName="NorthWind"
      enablePasswordRetrieval="false"
      enablePasswordReset="true"
      requiresQuestionAndAnswer="false"
      requiresUniqueEmail="false"
      passwordFormat="Hashed"
      minRequiredNonalphanumericCharacters="0"
      minRequiredPasswordLength="8"/>
    </providers>
    </membership>

    <roleManager enabled="true"
      defaultProvider="AspNetSqlRoleProvider">
      <providers>
    <clear/>
    <add
      name="AspNetSqlRoleProvider"
      type="System.Web.Security.SqlRoleProvider"
      connectionStringName="SqlConnectionString"/>
    </providers>
    </roleManager>
  </system.web>
</configuration>
```

## Conclusão

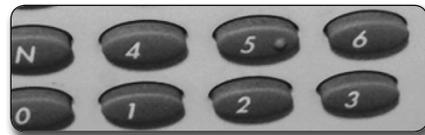
O *ASP.NET 2.0* veio repleto de novidades em segurança. Pudemos ver que juntamente com os controles de *Login*, é possível implementar segurança em nossas aplicações sem precisar incluir uma única linha de código! Fizemos tudo apenas configurando a segurança pelo *ASP.NET Configuration* e através dos controles de *Login*. Um abraço a todos e até a próxima! ●

### Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

[www.devmedia.com.br/clubedelphi/feedback](http://www.devmedia.com.br/clubedelphi/feedback)





# Out Sourcing Total

Preparados para nova era em qualidade de serviços, garantimos operações full-time dentro de uma gestão única desenvolvida pela nossa equipe para atuar preventivamente, somos a única empresa a operar por 25hs por dia garantindo disponibilidade total.

## Escalabilidade, Conectividade e Disponibilidade.

### CERTIFICAÇÕES

- ↳ Servidor Dedicado
- ↳ Gerenciamento
- ↳ Banda IP
- ↳ Cross Connection
- ↳ Link PAP
- ↳ Serviços compartilhados
- ↳ Sites
- ↳ E-mail
- ↳ Backup

MICROSOFT

LINUX

CISCO

ITIL

Data Center Próprio

## Segurança e certeza de bons negócios e serviços

[contato@localdatacenter.com.br](mailto:contato@localdatacenter.com.br)

+55(21) 3527-6510