**ATIVIDADE 8-1. EMITINDO E RESPONDENDO PARA AS MENSAGENS DE EVENTOS**

Nesta atividade, você irá aprender fazer o seguinte:

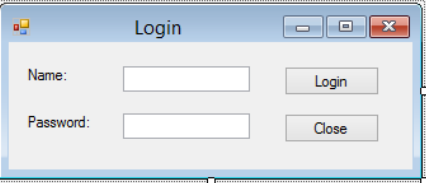
* Criar e dar os eventos de uma classe do servidor.
* Manipular eventos (handle events) de classes do cliente.
* Manipular eventos GUI (Graphics Users Interface).

**Adicionando e dando as mensagens de eventos na definição de classe**

Para adicionar e dar as mensagens de eventos em um arquivo de definição de classe, siga estes passos:

1. Inicie o Visual Studio. Selecione *File* -> *New* -> *Project*.
2. Escolha um projeto *Windows Forms Application*. Nomeie o projeto *Activity8\_1*.
3. Usaremos um padrão de nomes no formulário (form) que é incluso no projeto. Adicionando controles para o formulário (form) e mude os valores das propriedades, são listados na tabela abaixo. Você poderia completar o formulário (form) olhamos abaixo:

|  |  |  |
| --- | --- | --- |
| Object  (Objeto) | Properties (Propriedades) | Value  (Valores) |
| Form1 | Name  Text | frmLogin  Login |
| Label1 | Name  Text | lblName  Name: |
| Label2 | Name  Text | lblPassword  Password: |
| Textbox1 | Name  Text | txtName  (empty) |
| Textbox2 | Name  Text  PasswordChar | txtName  (empty)  \* |
| Button1 | Name  Text | btnLogin  Login |
| Button2 | Name  Text | btnClose  Close |



1. Selecione *Project* -> *Add Class*. Nomeie a classe para *Employee*. Abre o código da classe *Employee* no editor de código.
2. Em cima da declaração, adicionamos a seguinte linha de código para definir o evento LoginEventHandler delegate (delegar). Você irá usar isto no vento para trilhar os logins dos empregados (employee) na sua aplicação:

*public delegate void LoginEventHandler(string loginName, Boolean status);*

1. Dentro da declaração de classe, adicionamos a seguinte linha de código para definir o *LoginEvent* como um tipo *delegate*:

*public event LoginEventHandler LoginEvent;*

1. Adicione o seguinte método *Login* para a classe, com isso irá criar o *LoginEvent*:

*public void Login(string loginName, string password)*

*{*

*//Data normally retrieved from database.*

*if(loginName == “Smith” && password == “js”)*

*LoginEvent(loginName, true);*

*else*

*LoginEvent(loginName, false);*

*}*

1. Selecione *Build* -> *Build Solution*. Tenha certeza que não erros na janela *Error List*. Se tiver, então conserte-os, e então rode novamente.

**Recebendo Eventos na Classe do Cliente**

Para receber os eventos na classe do cliente, siga esses passos:

1. Abra a janela *Design* na form *frmLogin*.
2. Dê duplo clique no button (botão) *Login* para visualizar o evento *Click Event Handler* do button *Login*.
3. Adicionando o seguinte código na classe *Employee* (Empregado) com um *event handler* na classe *form*:

*private void btnLogin\_Click(object sender, EventArgs e)*

*{*

*Employee oEmployee = new Employee();*

*oEmployee.LoginEvent += new LoginEventHandler(oEmployee\_LoginEvent);*

*oEmployee.Login(txtName.Text, txtPassword.Text);*

*}*

1. Adicionando o seguinte método do *event handler* para o *form* (formulário) que obtém a chamada quando a classe *Employee* emite um *LoginEvent*:

*void oEmployee\_LoginEvent(string loginName, bool status)*

*{*

*MessageBox.Show($”Login status : { status }”;*

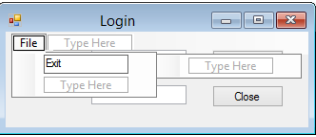
*}*

1. Selecione *Build* -> *Build Solution*. Tenha certeza que não erros na janela *Error List*. Se tiver, então conserte-os, e então rode novamente.
2. Selecione *Debug* -> *Start* para rodar o projeto.
3. Para testar tenha que certeza que o evento (*event*) *Login* está criado, entre com o nome no campo “*Login Name*” “Smith” e a senha no campo “*Password*” de “js”. Isto poderia fazer um gatilho no *status* do login como *true*.
4. Depois de testar o evento *Login*, feche o *form* (formulário), isso irá fechar o depurador.

**Manipulando eventos múltiplos com um método**

Para manipular múltiplos eventos com um método, siga esses passos:1.

1. Abra o *form* (*frmLogin*) na tela do formulário clicando duas vezes no nó *frmLogin* na Solution Explorer e escolha *View Designer*.
2. Do *Toolbox*, adicione um controle *MenuStrip* para o form (formulário). Clique onde diz “*Type Here*” e digite *File* no topo do menu e no sub-menu *Exit*, conforme a figura abaixo:



1. Adicionando o seguinte método para o manuseio no evento *click* do menu e o botão (*button*) *Close*:

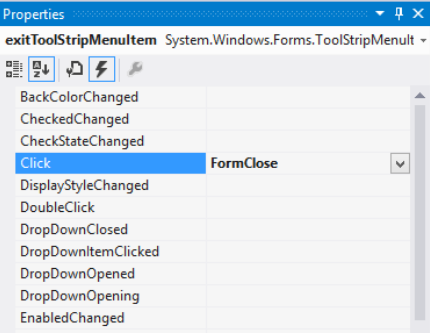
*private void FormClose(object sender, EventArgs e)*

*{*

*this.Close();*

*}*

1. Abra o *frmLogin* na janela *Designer*. Na janela *Properties*, selecione *exitToolStripMenuItem*. Selecione o evento *button (ligtning bolt*) no topo na janela *Properties* mostra os eventos do controle. Clicando no evento *drop-down*, selecione o método *formClose*, na figura abaixo:



1. Repita o passo 4 para o botão (*button*) *btnClose* no evento *click* para o método *FormClose*.
2. Expanda o nó do *Form1.cs* na janela *Solution*. Clique com o botão direito no nó *Form1.Designer.cs* e selecione *View Code*.
3. No editor do código, expanda a janela *Windows Form Designer* gerado no código da region. Procure pelo código listado abaixo. Este código é gerado pelo *form designer* por cima dos eventos por método *FormClose*.

*this.btnClose.Click += new System.EventHandler(this.FormClose);*

*this.exitToolStripMenuItem.Click += new System.EventHandler(this.FormClose);*

1. Selecione Build -> Build Solution. Tenha certeza que não erros na janela Error List. Se tiver, então conserte-os, e então rode novamente.
2. Selecione Debug -> Start para rodar o projeto.
3. Depois de testar, salve o projeto, e então saia do Visual Studio.