

GETTING STARTED

Visual Guard.Net



www.visual-guard.com

Table of content

ABOUT THIS DOCUMENT.....	1
Introduction.....	1
PRESENTING VISUAL GUARD.NET.....	2
Manage the security in a .NET application.	2
How does Visual Guard.Net work?	4
USING VISUAL GUARD.NET	6
Integrate Visual Guard.Net into the application.....	6
The integration of Visual Guard.Net into your application is completed in 4 steps.	6
Integrate the Visual Guard «runtime»	7
Add the necessary code to Visual Guard.....	8
Create a repository and declare the application.....	11
Define permissions in Visual Guard.....	13
Defining permissions sets, roles and users.	19
Permission sets.....	20
Permission sets allow you to gather permissions in a coherent group, then available to be attributed to a group of users. Each permission set is defined by a list of permissions....	20
The roles	21
The users	22
CONCLUSION.....	26

About this document

Introduction

Welcome to Visual Guard.Net's Getting Started Guide.

This document will take you through the main features of Visual Guard.Net. It explains step by step how to implement security to your .Net application.

This tour lasts less than an hour and does not require any specific expertise.

Do not hesitate to contact us if you have any questions concerning Visual Guard and its evaluation.

Minimum configuration

You need the following configuration to launch Visual Guard.Net and its sample application, Visual Guard Winform Sample:

Windows 98/2000./XP

Pentium 500 MHz

128 MB RAM

20 MB free on the hard drive

The installation of Visual Guard.Net's evaluation version (or full version) is required.

Audience

This guide is directed to developers and to Team leaders who wish to secure their .Net application.

This evaluation does not require any specific programming know-how. Nevertheless it is advised to know the basics of .Net development (VB.Net or C#) to understand some of the concepts in this presentation.

Presenting Visual Guard.Net

Manage the security in a .NET application.

Standard features to secure .Net applications are available in the .Net framework. They are complex to use and they require heavy coding and maintenance.

Moreover these features focus on securing the system and the access to the code. .Net does not allow a person without the appropriate permissions to execute a piece of the code nor does it allow this person to use system resources (resources systems) (printer, network ...).

The .Net framework does not offer any features to forbid/ avoid a user from accessing your application or a part of your application.

Take the example of an application which manages employees and purchase orders. On one hand you need to avoid that the Human Resources Manager modifies a purchase order. On the other hand you also want to avoid that a salesperson looks into a co-worker's private information.

The .Net framework mentions the concept of «Role»: a user can have a role in the application. The use of these roles is limited: they avoid that people execute a certain piece of code of the application (a class, a method ...). If this user attempts to execute this forbidden piece of code, the execution is interrupted and an error message is generated.

This solution is hardly maintainable in the long run especially as roles evolve in time.

Of course this method is very secure but it needs a rigorous organisation of the code as all the different needs have to be taken into account to secure an application. As often as these needs evolve, you have to update the organisation of the code. This is very quickly a complex and costly solution.

Moreover this solution is not user friendly. It is better not to have the option (of deactivating a button for example) than to have it, use it and receive an error message box.

To over go these limitations you can code inside the application itself. That will answer the ergonomic issues or the need to filter data but it will amplify the development - and mostly the maintenance - costs.

Visual Guard was especially created to complete the .Net framework and solve these problems. The tool allows you to manage users and their permissions easily and adapt the GUI according to them.

The advantages of Visual Guard

Visual Guard allows you to add to your application an authorisation process (Who is entitled to do what in the application) and an authentication process (user and password management).

... Just a few lines of codes are needed!

Visual Guard will make the security of your applications much more dynamic.

It is easy to define new types of users (roles) without having to modify the code.

Visual Guard has an administration console allowing user and permission management by an administrator which doesn't have to be a developer.

Powerful features will enable you to modify in depth the behaviour of the application according to the user profile (type)

The Security aspect is totally separated from the rest. Its maintenance is simplified and therefore its efficiency.

When using the Visual Guard framework you standardize authentication and authorisation features in all your application and centralize user management in one tool.

How does Visual Guard.Net work?

There are two steps: during the development (or maintenance) phase of the application and during the administration of the security.

You need

- To integrate the Visual Guard «runtime» in your application, which means adding a few assemblies as a reference in your project and initializing Visual Guard.Net with a few lines of code?
- To use the Visual Guard.Net administration console which is part of the setup of Visual Guard.Net?

During the development phase the developer defines the list of permission that will be granted to users.

For example: he can define permission (or rather a restriction here) to forbid creating or modifying the employees list. He will have to specify that the buttons “new” and “delete” need to be de-activated from the form “employees lists” and activate it in “read-only mode” all the fields in the form.

All this is managed with the administration console without adding any code to the application itself. We will see how in a later chapter.

During the security administration phase, the person in charge of the applications security is going to define the different users and their role in the applications.

For example, he can create the user «*John Smith*». The latter is in charge of HR, the administrator will therefore give him the “HR employee” role.

This role will have had to be created first in order to have a permission set associated to all HR employees.

This information is stored in a repository. When the user launches the application, he authenticates himself to Visual Guard.Net. The tool then loads the permissions associated to his role in the application. Visual Guard applies these permissions to the application dynamically. For instance Visual Guard will disable the buttons “new” and “delete” from the form “employees lists” when it opens.

We have now seen how Visual Guard works. Let’s review each step in detail.

Using Visual Guard.Net

Integrate Visual Guard.Net into the application

This getting started is based on a WinForm application. If you want to integrate Visual Guard in a WebForm application, you must consult the developer documentation or the code of the WebForm sample project installed with the product (menu *Start > Novalys > Visual Guard for .Net 2.6 > Sample > Sample Solution*).

The integration of Visual Guard.Net into your application is completed in 4 steps.

- Adding the Visual Guard «runtime» to the Visual Studio project of your application.

- Inserting the code which will identify the user and activate the security.
- Creating a Visual Guard repository and declaring your application through the administration console.
- Generating Visual Guard configuration parameters. These parameters will be stored in 2 configuration files «*VisualGuardConfiguration.config*» and «*VisualGuardConfiguration.exe.config*».

Let's see these steps one by one. We will use the sample application installed in the «*Sample\VB*» repository.

If the development part is of less interest to you or if you do not wish to see the integration details, please jump to page 12, the chapter “Defining the permissions in Visual Guard.Net”.

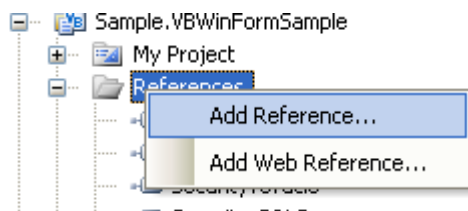
Integrate the Visual Guard «runtime»

You need to open the project «*Visual Guard VB Sample.vbproj*» in Visual Studio under «*VB Sample*».

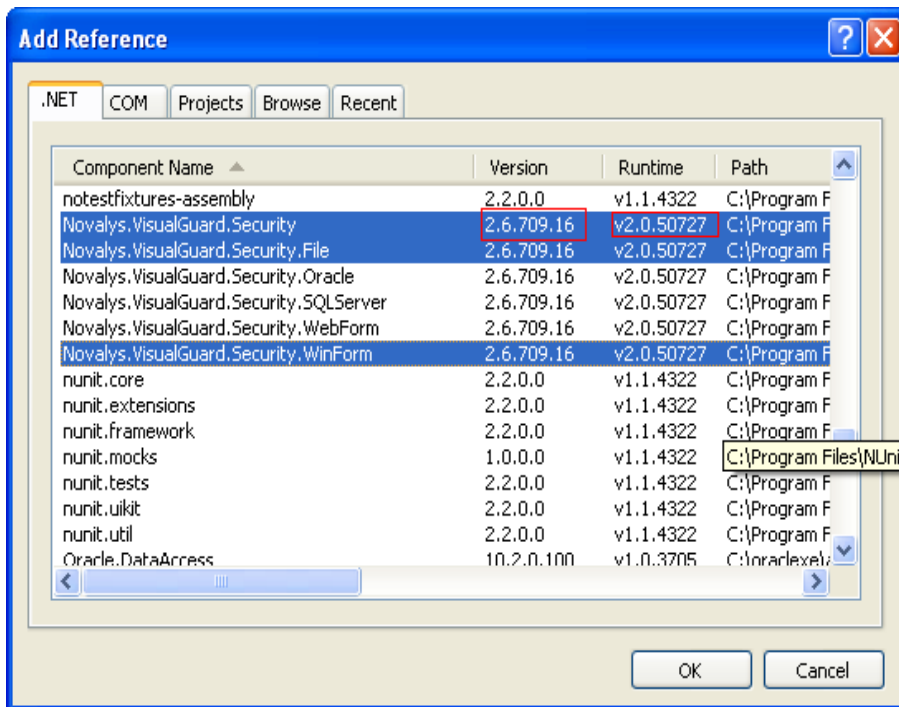
It is necessary to add a reference to the Visual Guard.Net's assemblies to use the tool in your application. Here is a description of these assemblies:

- «*Novalys.VisualGuard.Security*» assembly contains main Visual Guard's classes. This assembly must always be referenced.
- «*Novalys.VisualGuard.Security.File*» assembly contains classes needed to access to a repository based on file. It must be referenced only if you use this type of repository.
- «*Novalys.VisualGuard.Security.Oracle*» assembly contains classes needed to access to a repository stored in an Oracle database (8i or higher). It must be referenced only if you use this type of repository.
- L'assembly «*Novalys.VisualGuard.Security.SQLServer*» assembly contains classes needed to access to a repository stored in an Microsoft SQLServer database (2000 or higher). It must be referenced only if you use this type of repository.
- «*Novalys.VisualGuard.Security.WebForm*» contains classes needed for ASP.Net applications. This assembly must be referenced only if you integrate Visual Guard in an ASP.Net WebSite or WebService.
- «*Novalys.VisualGuard.Security.WinForms*» contains GUI classes for Winform applications (Authentication form, password modification form,...) . This assembly is needed if you use at least one of its classes. You can use your own forms if you prefer.

To add the reference you must select the item «References» and right-click to select the option «Add reference».



In the dialog box, select Visual Guard's 3 assemblies, click on "Select" and "OK".



Note: If the list displays more than one version of Visual Guard assemblies, you must select the assembly corresponding to the version of the runtime used by your project and the version 2.6.x.x of Visual Guard assemblies.

If these assemblies do not appear in the assemblies list, you must:

- Click the "Browse" tab.
- Browse to the "Bin" directory placed under the Visual Guard installation directory (by default "C:\Program Files\Novalys\Visual Guard 2.6 For .Net\Bin"). Select the directory corresponding to the version of your framework (1.1 for .Net 1.1, 2.0 for .Net 2.0).
- Select the assemblies, click the "Open" button then click the "OK" button.

Add the necessary code to Visual Guard

Once the references to the assemblies are added, we can add the code to identify a user and secure the applications' objects.

There are two main classes in Visual Guard:

- The class «*Novalys.VisualGuard.Security.VGSecurityManager*» which is Visual Guard's main class. It is used to centralize the actions necessary to the inner workings of Visual Guard.
- The class «*Novalys.VisualGuard.Security.WinForms.VGLoginForm*» ; the default login window in Visual Guard. It is used to identify a user while using Visual Guard. It is not compulsory as you can use your own window and call the login window of «*VGSecurityManager*» yourself.

As a general rule, the security loading of Visual Guard has to be done before any other code. This will guarantee that Visual Guard will apply the security actions of all the objects of the applications correctly.

Here is an authentication example done by Visual Guard. It is coded in the click of an "ok" button of an authentication form (this code needs to add reference to "*Novalys.VisualGuard.Security*" and "*Novalys.VisualGuard.Security.WinForms*" namespaces):

```

Dim state As VgAuthenticationState =
VgSecurityManager.Authenticate(userTextBox.Text, passwordTextBox.Text)
If state.IsFailed() Then
Me.DialogResult = DialogResult.None
If state.IsCanceled() Then Return
If state.IsCredentialInvalid() Then
If state.IsLastBadLogin() Then
MessageBox.Show("User or password invalid" & _
Environment.NewLine & "The next bad login will lock your account.")
Else
MessageBox.Show("Invalid user or password")
End If
ElseIf state.IsUserNotAuthorized() Then
MessageBox.Show( _
"You are not authorized to log on to this application")
ElseIf state.IsUserAccountExpired() Then
MessageBox.Show("your account is no more valid. " & _
Environment.NewLine & "Contact your administrator")
ElseIf state.IsUserAccountNotYetAvailable() Then
MessageBox.Show("your account is not yet available.")
ElseIf state.IsUserAccountLocked() Then
MessageBox.Show("your account is locked. " & _
Environment.NewLine & "Contact your administrator.")
End If
Else
Me.DialogResult = DialogResult.OK
If state.IsLastGraceLogon() Then
MessageBox.Show("Your password is not enough secure." + _
"You must change your password")
' Use a custom form the Visual guard form to change
' password. You can replace it by VgChangePasswordForm
' located in Novalys.VisualGuard.Security.WinForms
Dim form As ChangePassword = New ChangePassword
form.ShowDialog()
Else
If Not state.IsPasswordSecure() Then
If MessageBox.Show( _
"Your password is not enough secure. " + _
Environment.NewLine & "Do you want to change it?", "", _
MessageBoxButtons.YesNo, MessageBoxIcon.Question, _
MessageBoxDefaultButton.Button1, _
MessageBoxOptions.DefaultDesktopOnly) = DialogResult.Yes Then
' Use a custom form the Visual guard form to change the
' password. You can replace it by VgChangePasswordForm
' located in Novalys.VisualGuard.Security.WinForms
Dim form As ChangePassword = New ChangePassword
form.ShowDialog()
End If
End If
End If
End If

```

If you prefer using the login form provided by Visual Guard, here is an example of the code you need to add in the function "Main" of your application.

```

Dim login As New Novalys.VisualGuard.Security.WinForms.VGLoginForm
If login.ShowDialog() = DialogResult.OK Then
    login.Dispose()
    Application.Run(New MDIForm)
End If

```

For VB.Net application, when the application framework is enabled, you must add the following code in the event Startup of your application (open the Project properties designer then click the button “View application events”)

```

Imports Novalys.VisualGuard.Security.WinForms
Imports System.Reflection
Imports Microsoft.VisualBasic.ApplicationServices
Namespace My
    Partial Friend Class MyApplication

```

```

        Private Sub MyApplication_Startup(ByVal sender As Object, ByVal e As
StartupEventArgs) Handles Me.Startup
            Dim form As VGLoginForm = New VGLoginForm()
            If form.ShowDialog() <> DialogResult.OK Then
                e.Cancel = True
                Return
            End If
        End Sub
    End Class

```

```
End Namespace
```

In some cases, you might not want to use the authentication mechanism provided by Visual Guard but your own (use the login Windows with *WindowIdentity* for example). In this case you can use the method *VGSecurityManager.LoadSecurity*. This method checks the authorization to access the application and loads the permissions. Here is an example of code when you are using the login Windows to identify the user:

```

Dim state As VGAuthorizationState
state = VGSecurityManager.LoadSecurity _
    (System.Security.Principal.WindowsIdentity.GetCurrent())
If state.IsFailed() Then
    If state.IsUserNotFound Then
        MessageBox.Show( _
            "Your are not declare in the security repository")
    ElseIf state.IsUserNotAuthorized Then
        MessageBox.Show( _
            "Your are not authorized to log on to this application")
    End If
Else
    Application.Run(New MDIForm)
End If

```

If you only need to secure the forms of your application, you have finished the integration of Visual Guard! You can now go to the next chapter and learn how to create the repository. Visual Guard detects automatically the instantiation of the forms and automatically applies security permissions.

But if you want Visual Guard to secure in other type of classes, you need to add the following code in each of these objects.

- Add the interface *Novalys.VisualGuard.Security.VGISecurable* to your classes. This interface does not have any methods to implement but Visual Guard uses it to determine that this class has to be secured and will display them in the list of classes in the Visual Guard console when you need to add security actions.
- Add a call to *VGSecurityManager.SetSecurity* method in your object's constructor. Visual Guard will secure it. This call is generally added at the end of the constructor's script (after *InitializeComponent* for a graphical component). Here is an example of the script you need to add:

```
Novalys.VisualGuard.Security.VGSecurityManager.SetSecurity(this)
```

We have finished the integration of Visual Guard in the application; if you wish to have more examples about the integration, you can have a look at the sample application. You can open the solution containing the sample from the Windows Start menu and navigate to the solution as follows:

Start > Programs > Novalys > Visual Guard for .Net 2.6 > WinForm Sample > Visual Guard Sample Solution

Let's now see how to create the repository containing Visual Guard's security data.

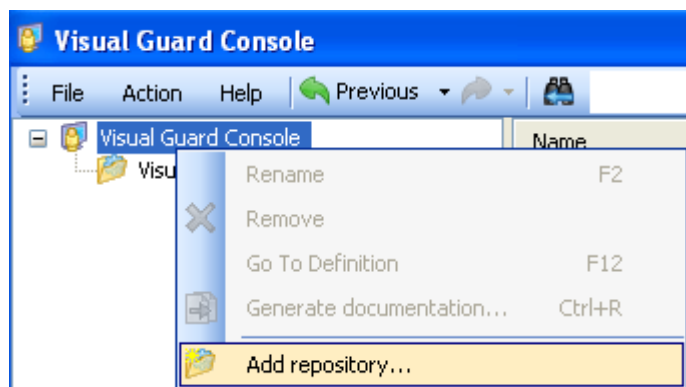
Create a repository and declare the application.

In Visual Guard's repository, you will find information about the users of the application, the roles available, the permissions and the security actions implied when securing the application.

Everything is managed by the Visual Guard console. You can launch it from the Windows Start menu and navigate to the solution as follows:

Start > Programs > Novalys > Visual Guard for .Net 2.6 > Visual Guard Console

Select the item Visual Guard Console and right click to select the option «*Add repository*» The Wizard opens.



On the second page of this wizard choose «*Create a new empty repository*» and on the next page select the item «*File*». This will store the repository as files.

On the next page, click on the button «...» to indicate which file the repository will be stored in.

For the demo, create a folder «*Visual Guard*» in «*My documents*» then select it. Click on the button «*Next*».

The next page allows you to select the mode of authentication supported by the repository. Visual Guard supports three authentication modes:

- **Visual Guard authentication mode:** this mode uses the built-in authentication process to authenticate an user. The user and the password are stored in the Visual Guard repository. This mode allows you to easily add an authentication mechanism to your application without to add new component in your infrastructure.
- **Windows authentication mode:** this mode uses Windows accounts to authenticate the users. In this mode, The user do not have to provide user/password information to be authenticated, Visual Guard loads the authorisation for the current Windows identity. This mode is very secure and tightly integrated with the Windows security.
- **Database authentication mode:** this mode uses the database accounts and authentication mechanism to authenticate the user. The accounts must be defined in the database and must have access to the Visual Guard repository data. During the authentication process, the user must provide its database user id/password. Visual Guard will try to open a connection with this credentials and load authorizations. If the process fails, the user is not authenticated. This mode is enabled only if the repository is stored in a database.

For the sample, you must select the Visual Guard authentication mode only then click on the button «*Next*».

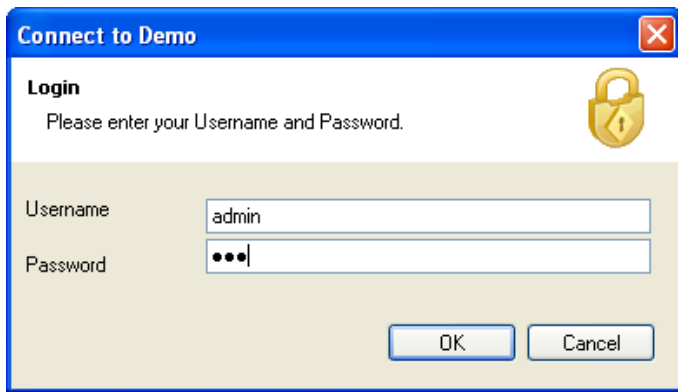
The following page enables you to define, who will be the default master administrator, the one able to administrate this repository. Once the wizard completed, Visual Guard will create this user in the repository and will give him administrator rights.

For the demo you can keep the name of the default user («*Admin*») and choose «*pwd*» as password. Be careful not to forget this password otherwise you will not have access to the repository any more.

You can give «*Demo*» as name of the repository.

Click on «*Finish*» Visual Guard will create your repository.

Now the repository is created, you need to connect to it. Click on the «*Demo*» item (the item corresponding to the repository created) and right-click on the option «*Connect...*».



You have to type in the user name and the password you determined in the Wizard. (*Admin/pwd*) and click on “OK”.

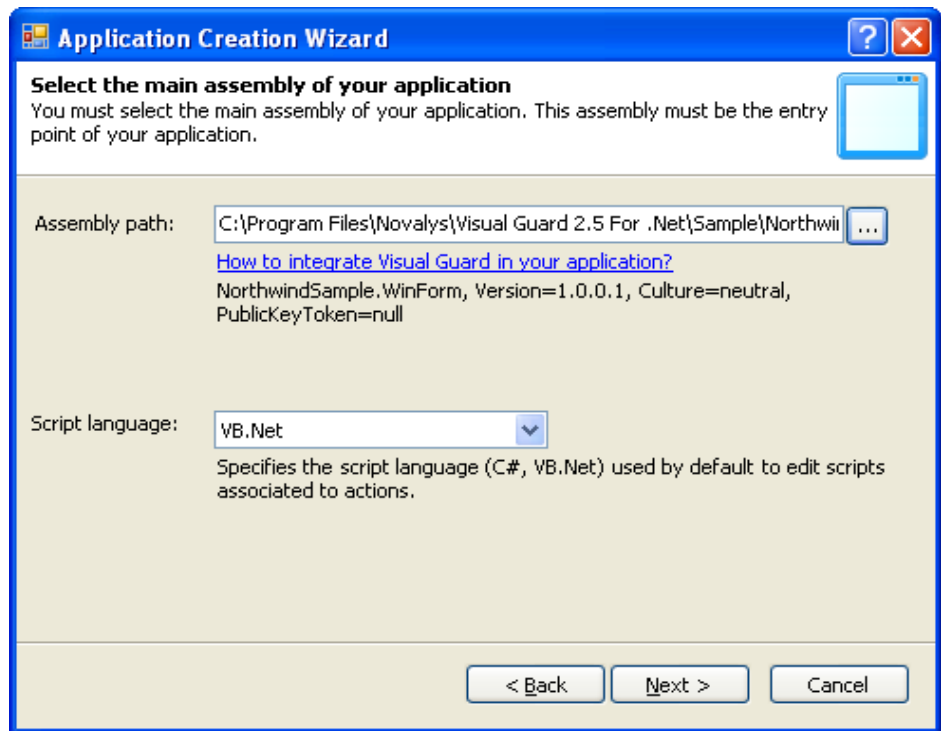
After connecting, Visual Guard launches a wizard. This wizard enables you to declare the type of the application and the location of the application.

On the first page, you must indicate the type of the application. You must select the “WinForm” option then click the “Next” button. Confirm that the Visual Guard runtime is integrated then click the “Next” button.

In the next page, you can indicate your application’s executable file.

For the demo, select the file “*NorthwindSample.WinForms.exe*” in the directory “*C:\Documents and Settings\All Users\Application Data\Novalys\Visual Guard for .NET\2.6\Sample*”. Then click the “Next” button.

(For some users, this folder is hidden. In this case, copy the specified path in the dialog box then click the 'Open' button in order to display this folder).



The next page allows you to define default options for your application. You must select the option “Allow anonymous session” when your application does not require a user identification. In this case, a user can access to the application without providing any user or password. Visual Guard will grant automatically the role “Anonymous” to this user.

When you select, the option “*Enable Default Role*”, Visual Guard will automatically grant the role “*Default*” to all users created in this repository. This option is essentially used in ASP.Net application, when the application allows membership subscription. In this case, the user provides identification information then Visual Guard will automatically create the user in the repository and grant the default role to this user.

In your case, you should disable this option then click the “*Finish*” button.

Click “*OK*” when Visual Guard asks for confirmation to replace old configuration files then click “*OK*” when Visual Guard indicates that these configuration files will have to be deployed with your application.

We have finished declaring the application. We can now define the authorisations, which will be attributed to the users.

Define permissions in Visual Guard

In Visual Guard, permission can authorize or restrict the use of a part of your application.

Here are some examples of permissions: forbidding the modification of the “*Sheet Employee*”, authorizing to administrate the system table, limiting the employee list to one country only.

Once all permissions of an application have been defined, they are gathered into permissions sets. A set of permission is a coherent group of permissions, which can be associated to a role. For example, the set of permission «Human Resources permissions» contains the following permissions:

- Deactivate the capability of modifying the client’s information.
- Deactivate the capability of modifying the product list and the products categories.
- Hide information from a purchase order.
- ...

A permission corresponds to a set of actions Visual Guard will carry out in the application (deactivate a menu, filter a list, hide a field). The developer is in charge of defining the permissions and the list of actions.

In Visual Guard, there are two kinds of action:

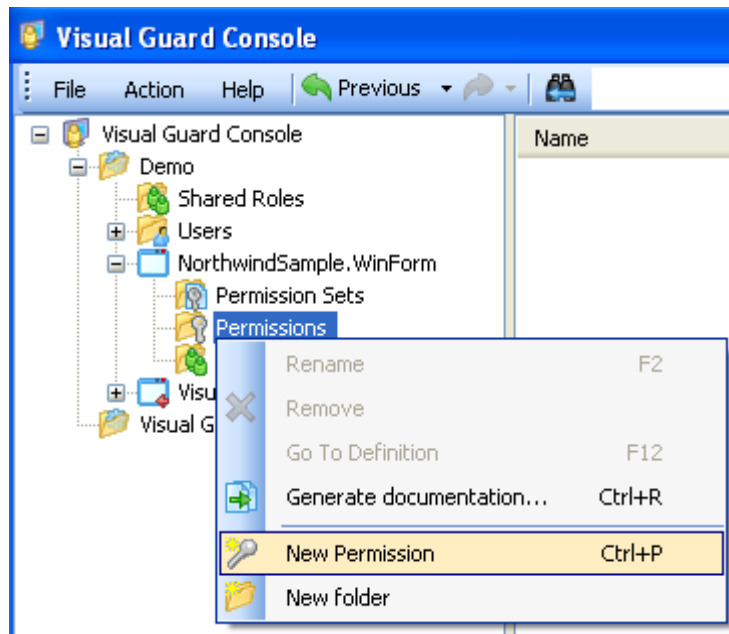
- *Properties actions* : they allow to create an action which will modify the properties of the objects of your application (for example in the form “Customer» you can choose to set “false” the property “enabled” of the button “Delete button”).
- *Script actions*: they allow writing a script (in Visual Basic or in C#) which will be dynamically executed in the application.

For each action, you must define:

- What class the action will be performed for (i.e. the form “customer”).
- What time the action must be performed at. Should it be executed when the object is secured (after the constructor for forms, or when the method *SetSecurity* is called) or at a given event of the class (*Load*, *Click*).
- The properties of the action themselves.

Let’s see now how to define the permission “*disable employee edition*”. To do so, you have to:

- Select the item “*Permissions*” under the application “*NorthwindSample*” and right-click to select the option “*New>Permission*”

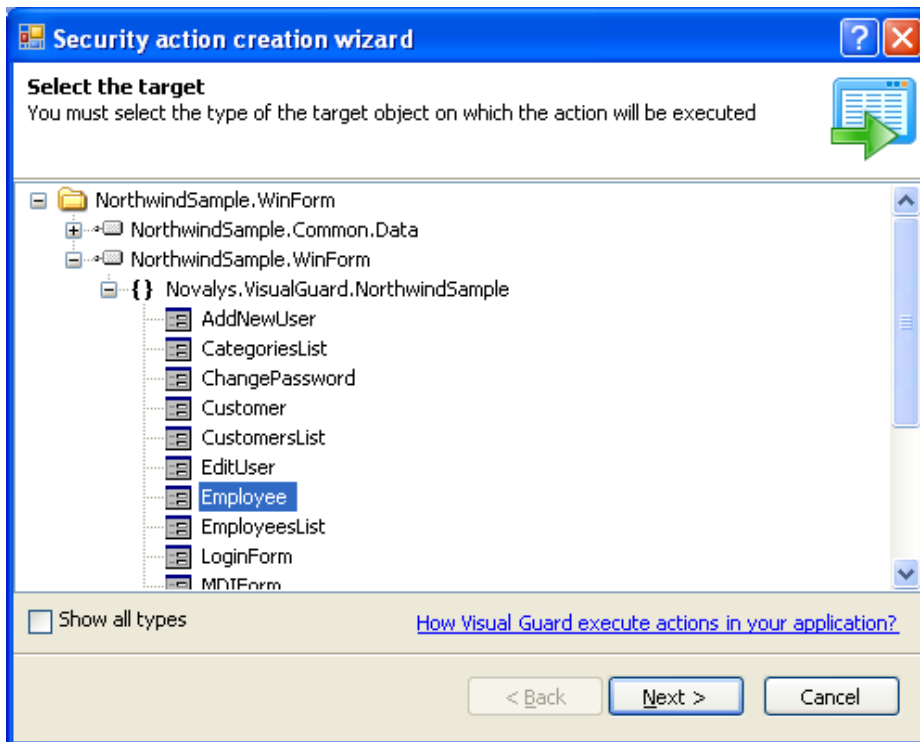


- Rename the item with the “*disable employee edition*”.

We will now create an action to disable the buttons “*delete*” and “*Save*” of the client’s form (the form *Employee*) and put all the fields of the form in “*ReadOnly*” mode.

To do so select the permission created and right-click to select the option “*New Properties Action*”.

The creation wizard opens. You can select the class which the action will be performed for in the first page; select the form “*Employee*” in “*NorthwindSample.WinForms*” and the namespace “*Novalys.VisualGuard.NorthwindSample*”.



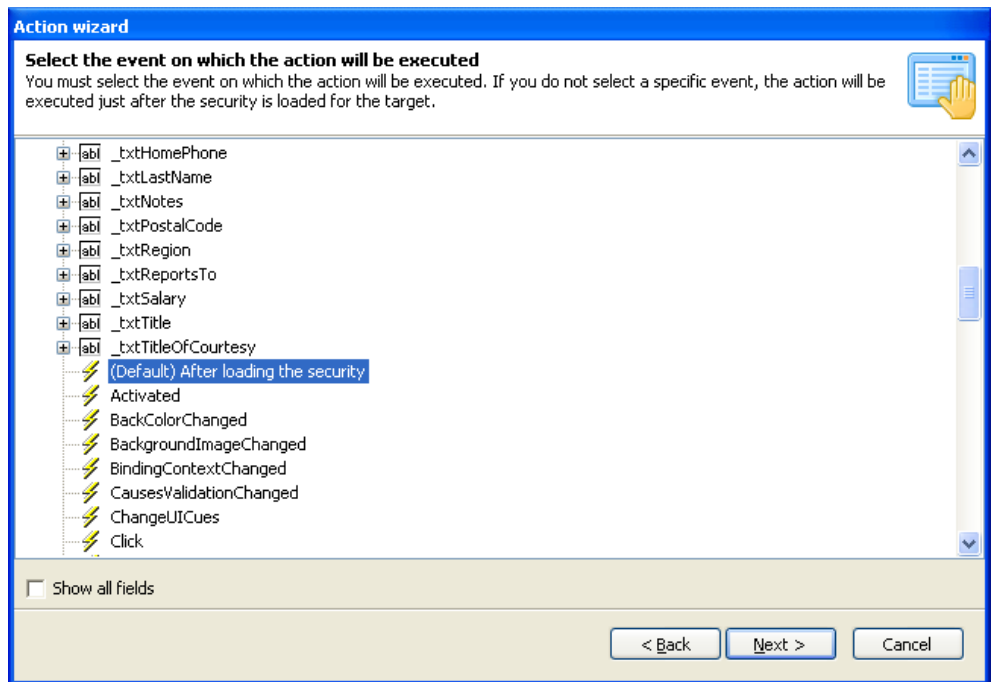
Then click on the button “Next”. This second page allows you to indicate what time the action will be performed at. In most cases, you will want the action to be executed as soon as possible:

If it is a form, Visual Guard will intercept its creation automatically and the security will be activated immediately after the execution of the constructor.

If it is a class which is not automatically intercepted by Visual Guard, the action will perform during the call to the *SetSecurity* method (this call must generally be done at the end of the object’s constructor).

In more specific cases, you will have to choose an other event of the class. For instance, an action can be executed when the user clicks on a button, after uploading the data, etc.

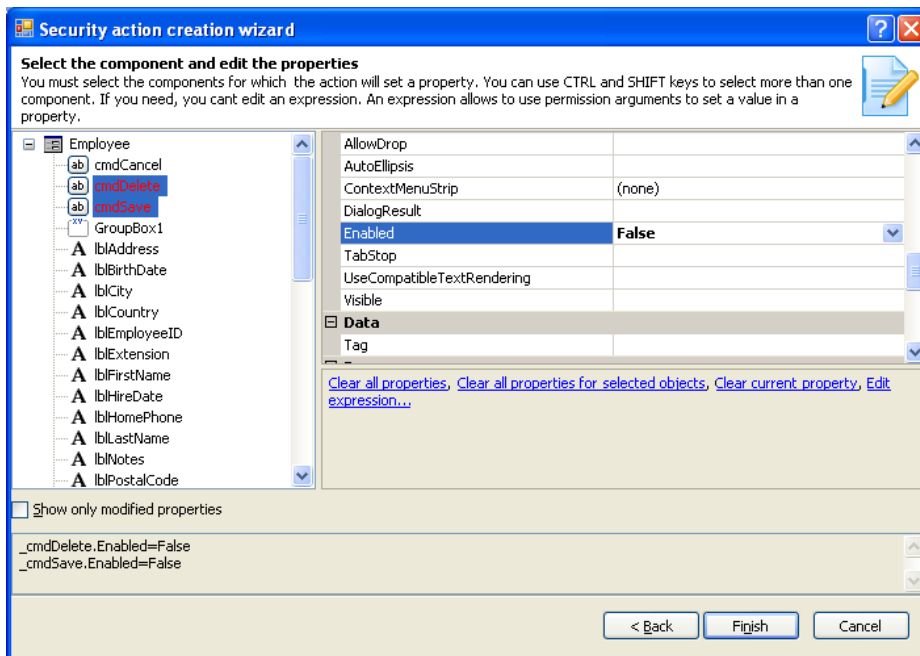
In our case, we are going to apply the action when the security is loaded.



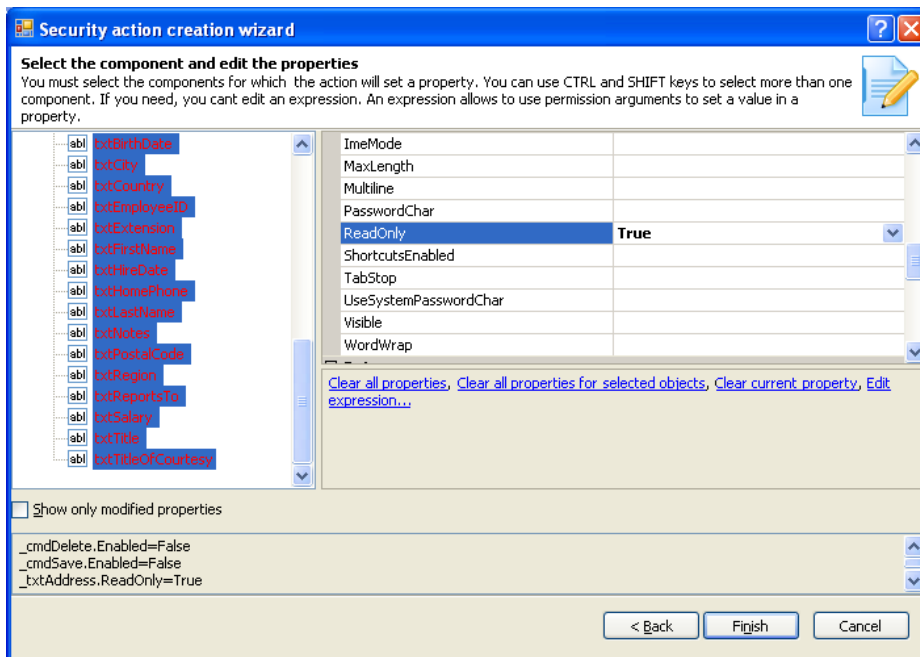
Now let's click on "Next" and go to the last page of the wizard. Here we are going to define the properties modified by the action. We are going to modify the buttons "cmdSave" and "cmdDelete"'s property «Enabled»:

Select the items "cmdSave" and "cmdDelete" in the treeview (use the CTRL key to selection both items at the same time).

In the property editor, modify "Enabled" to "false".



We are now going to put all the form's fields in "ReadOnly" mode. To do so, you have to select all the items, which begin by *txt* and modify the property «ReadOnly» to "true".



Notice that the items of the tree view have turned red in order to show that their properties have been modified. You can also:

- Cancel all modifications by clicking on «*Clear all properties*»
- Cancel the properties of the objects you selected by clicking on “*Clear all properties for selected objects*”.
- Cancel the value of the property you just selected in the property editor by clicking on “*Clear current property*”.
- Edit an expression. The option enables you to create an expression based on permission arguments. For example, if you create a permission with an argument containing a country, you can use this country to set the filter of a dataview.

We have now finished to creating the action. You can click on the button “*Finish*».

The permission we created contains one action only. It is nevertheless very easy to attribute other actions to it. For example, we could create an action to disable the button «*Delete*» from the Employees list’s form.

Now that we have created our first permission, let’s see how to define users and allow them to use the application.

Defining permissions sets, roles and users.

Permission sets

Permission sets allow you to gather permissions in a coherent group, then available to be attributed to a group of users. Each permission set is defined by a list of permissions.

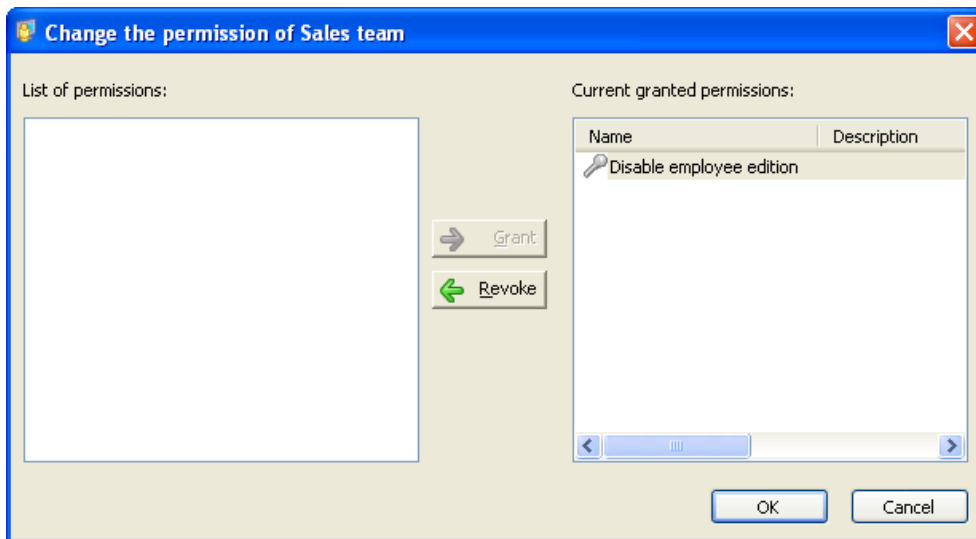
You can associate a permission set to another permission set. For example a permission set defined for the “*Sales Team*” can be used with the permission set “*Sales manager*” and the permission set “*Sale Person and Sales Assistant*”

This allows more flexibility with the permission management especially with complex applications with a high number of users.

Let’s create a permission set. Click on the item which corresponds to the application «*NorthwindSample*», right-click and select the option «*New Permission set*».

You can rename the permission set “*Sales Team*”. Right-click and select “*Edit permission list...*” to include the list of permissions.

In the dialog box select “*Disable employee edition*” and click on the button “*Add*”.



Click on the button «OK»; we have completed our permission set.

Note that it is possible to *drag-and-drop* the permissions on the permissions set

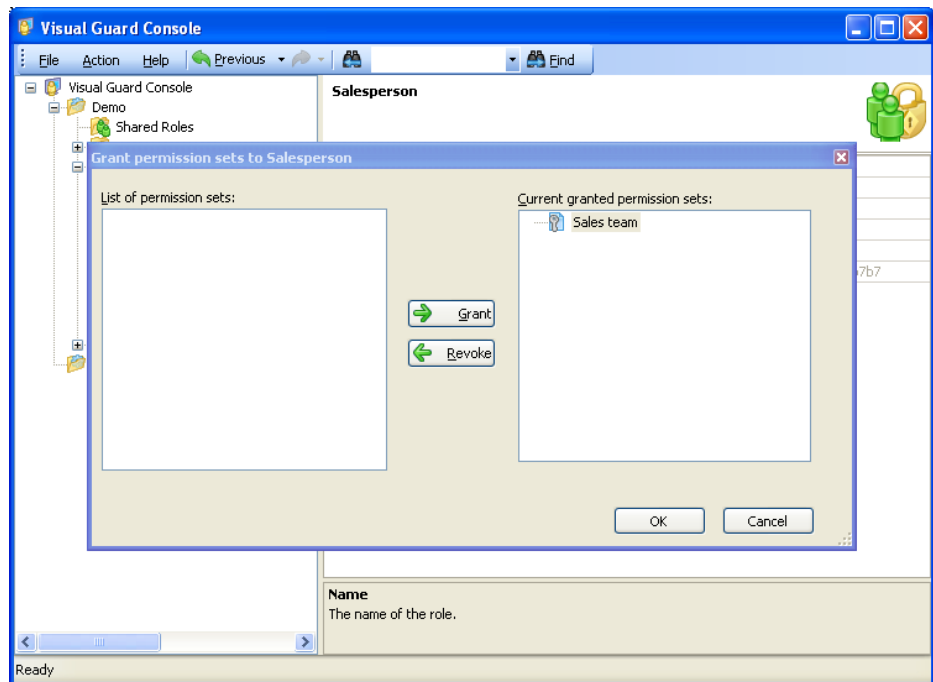
The roles

A role in Visual Guard is when you give a group of users the same rights. For example: Human Resources Manager, Sales Manager, Administrator...

These Visual Guard roles are compatibles with the notion of code access security of the .Net framework (by using the class "*PrincipalPermission*" or the attribute "*PrincipalPermissionAttribute*").

Let's create a role "*Salesperson*"; select the item "NorthwindSample" (which corresponds to your application), right-click and select "*New Role*".

Rename the role to "*Salesperson*" then right-click this item and select the option "Change permission set..." and grant "*Sales Team*" to the role then click "OK".



We have defined the role «*Salesperson*». This role will have the «*Sales team*» permission set.

Let's also create a role «*Administrator*». It won't be associated to any permission set. This means the user who has this role won't have any limitations in accessing the application. If the application was developed so that everything is secured to the maximum, the user «*Administrator*» will have the minimum authorisation in the application.

All we need to do now is define a user and associate him to the role «*Salesperson*» in the application.

The users

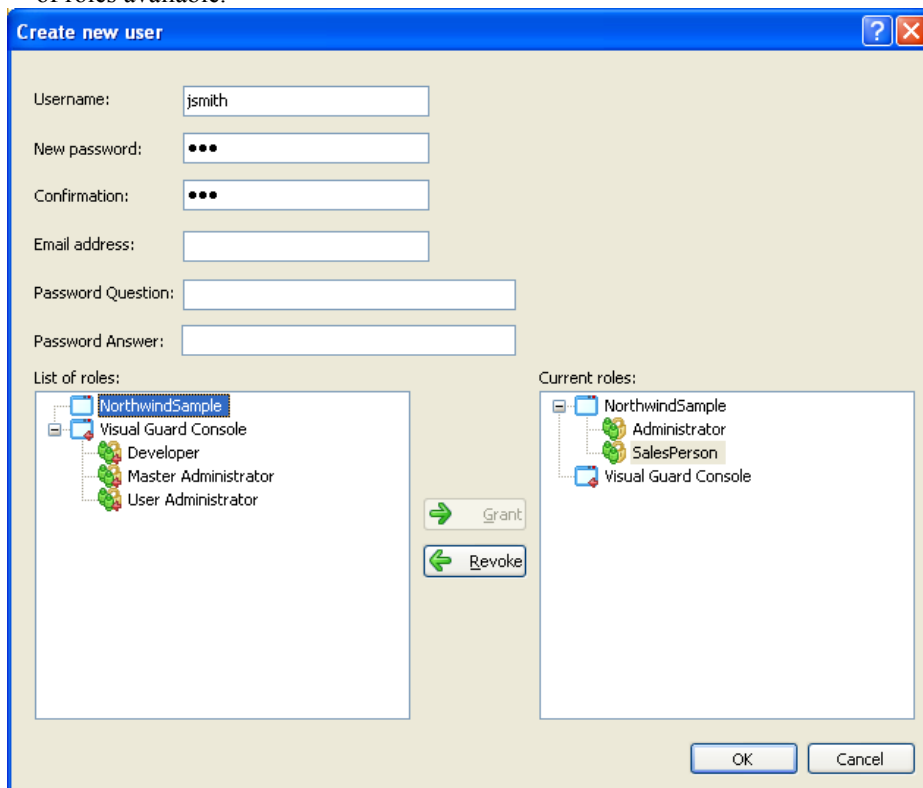
A user in Visual Guard is defined independently from the application. He can have access to one of applications declared in the repository or more. For a user to have access to an application, you have to associate him to the appropriate role in this application.

Visual Guard can manage the passwords of the users but it is possible to use your own authentication mechanism: for example if you wish to use the users created in the Windows domain or the accounts in your database to identify the users of the application.

In this case, you should define the users in the repository and grant them to a role without using the password management (to use the authorization mechanism without the authentication you need to use the method *VGSecurityManager.LoadSecurity* and not *VGSecurityManager.Authenticate*).

Each user can have one or more roles in the same application. In this case the user selects one during the authentication (this is what happens by default regarding the standard authentication window *VGLoginForm*), or the permission set of the user will contain all the permissions set granted to the roles.

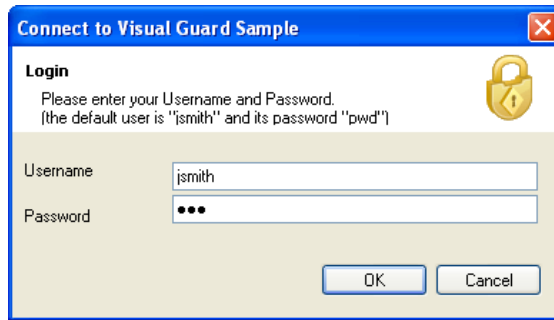
We are now going to define a user who will be able to access the application either as a Sales person or an administrator. The login form used by the application is the form provided by default by Visual Guard. It allows the user to choose a role from the list of roles available.



Select the item «*Demo*», which corresponds to our repository, right click then select the option “*New Visual Guard User*”. Type the username “*jsmith*” and the password “*pwd*”. To grant roles to the user, you must double-click on the item “*Salesperson*” and “*Administrator*” to grant these roles to “*jsmith*”.

Notice there are also roles for the application «*Visual Guard for .NET*». Indeed it is possible to grant to this user the permissions to use the Visual Guard console.

We can now test the user «*jsmith*» in the application. You need to launch the application. You can use the Start menu and run the application from «*Novalys > Visual Guard for .Net 2.6 > WinForm Sample > Visual Guard Sample*». The application displays the login form of Visual Guard. Type in «*jsmith/pwd*» as user and password.



Connect to Visual Guard Sample

Login

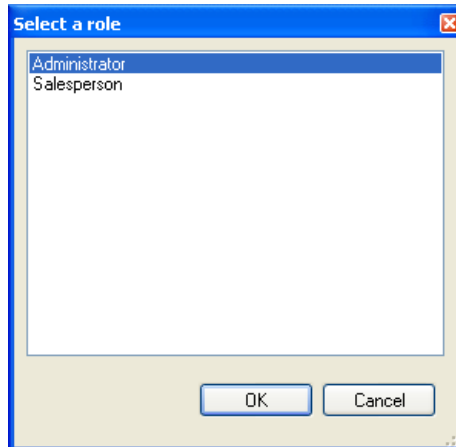
Please enter your Username and Password.
(the default user is "jsmith" and its password "pwd")

Username:

Password:

OK Cancel

The following dialog box will open up and display the roles granted to this user.

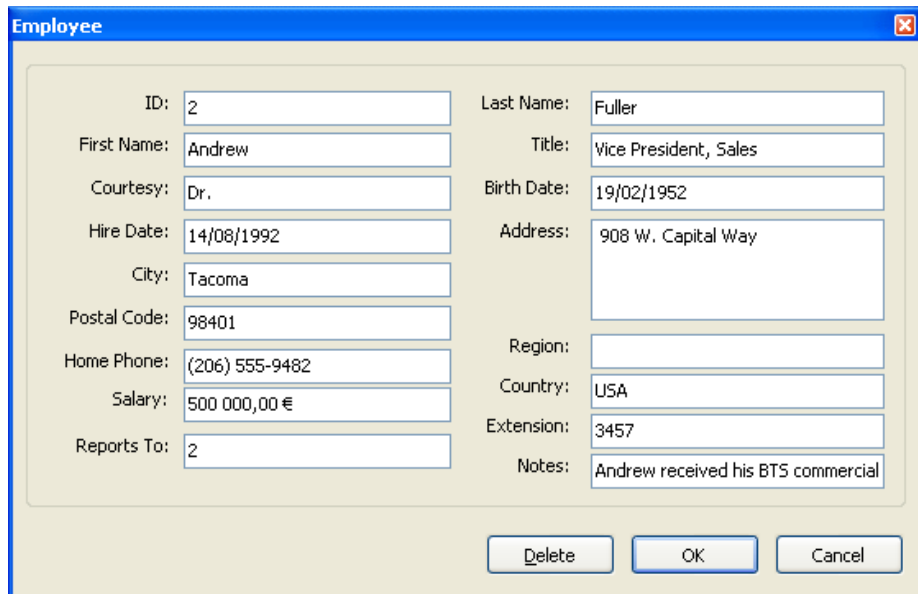


Select a role

- Administrator
- Salesperson

OK Cancel

You can choose the administrators role which has no restrictions. Open the list of Employees using the menu *"File>Employees"*. Double-click on one of the employees to display his form.

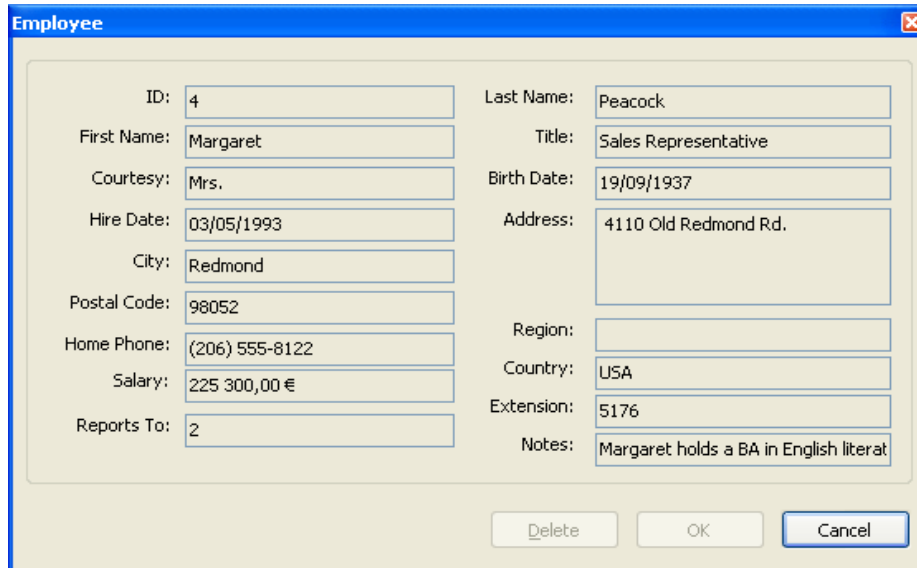


Employee

ID:	<input type="text" value="2"/>	Last Name:	<input type="text" value="Fuller"/>
First Name:	<input type="text" value="Andrew"/>	Title:	<input type="text" value="Vice President, Sales"/>
Courtesy:	<input type="text" value="Dr."/>	Birth Date:	<input type="text" value="19/02/1952"/>
Hire Date:	<input type="text" value="14/08/1992"/>	Address:	<input type="text" value="908 W. Capital Way"/>
City:	<input type="text" value="Tacoma"/>	Region:	<input type="text"/>
Postal Code:	<input type="text" value="98401"/>	Country:	<input type="text" value="USA"/>
Home Phone:	<input type="text" value="(206) 555-9482"/>	Extension:	<input type="text" value="3457"/>
Salary:	<input type="text" value="500 000,00 €"/>	Notes:	<input type="text" value="Andrew received his BTS commercial"/>
Reports To:	<input type="text" value="2"/>		

Delete OK Cancel

The user has no restrictions hence the buttons «OK» and «Delete» are activated and the data in each field is modifiable.



The screenshot shows a window titled "Employee" with a blue border and a close button in the top right corner. The window contains a form with the following fields and values:

ID:	4	Last Name:	Peacock
First Name:	Margaret	Title:	Sales Representative
Courtesy:	Mrs.	Birth Date:	19/09/1937
Hire Date:	03/05/1993	Address:	4110 Old Redmond Rd.
City:	Redmond	Region:	
Postal Code:	98052	Country:	USA
Home Phone:	(206) 555-8122	Extension:	5176
Salary:	225 300,00 €	Notes:	Margaret holds a BA in English literat
Reports To:	2		

At the bottom of the window, there are three buttons: "Delete", "OK", and "Cancel". The "Delete" and "OK" buttons are disabled (greyed out), while the "Cancel" button is active (blue border).

Now close the application, launch it again using «jsmith/pwd» but now select the role «Salesperson». Open the employees' list (menu *File>Employees*) and click on an employee. The form appears as «ReadOnly».

As it was mentioned in the action created above, the buttons «Delete» and «OK» are deactivated and the fields are «ReadOnly».

Conclusion

Congratulations! You have completed Visual Guard's guided tour.

As you have noticed yourself, Visual Guard is especially designed for the enterprise level management of the security. It offers a common repository for all your applications. It simplifies user and authorisation management with its administration console and the dynamic authorisation definition.

Visual Guard separates the development side of user permissions and the management side. The evolution of the application is therefore facilitated. You do not have to redefine each time the security strategy of your application.

If you wish to further your training on Visual Guard, you can create several more permissions and roles for the application «Visual Guard Sample».

You can now integrate Visual Guard in one of your applications to test it. If you have an evaluation version you have a maximum of 4 users in a repository.

Concerning the integration of Visual Guard in an application you can look at the code of the sample application provided with Visual Guard.

This manual is about the getting started of the following software:

Visual Guard

IDDN.FR.001.050019.01.S.P.2000.000.10600

The instructions in this document are for information only and can be subject to change without notice.

No part of this document may be reproduced, transmitted, copied in any way or by any means – electronic, mechanical, photocopying or recording, optical, chemical or else - for any purpose without the express written permission of
NOVALYS

While every effort has been made to ensure the accuracy of all information in this document, NOVALYS SA assumes no liability.

NOVALYS, Visual Expert, Visual Guard, PB-LINK are trademarks of NOVALYS S.A. All others names are for reference only and are the property of their respective owner

Copyright © 2005 NOVALYS S.A. All rights reserved

NOVALYS S.A

41/43 rue Paul Bert

92 100 Boulogne Billancourt

France

Phone number: (+33) 1.41.31.82.82

Fax: (+33) 1.41.31.82.90

Visit our web site: <http://www.visual-guard.com>