Login Management Framework

Contents

Introduction

An application may need to control which users may access which parts of the application. Usually a user database is used to record user credentials and permissions. It is very simple to add web page security. See chapter "Web Page Security" of http://www.limnor.com/support/webDatabaseProgramming.pdf.

A complete security system involves more issues. There must be a secure way to store the user passwords in the database; to allow a user to modify his/her password; to provide a secure way to reset user password in the case of forgetting password. The application needs to provide a login process to check user credential and get user permissions. It uses the user permission information to allow and deny accessing of different parts of the application. It may need to provide a log off function; on logging off the application should close all parts of the application, which require access permissions.

Implementing such features properly may be tedious and error prone, resulting in a less secure application. Limnor Studio provides a Login Manager component and a login management framework to make it simple to add access control to an application. The framework properly implements security requirements and provides a programming interface for you to add protection to applications easily. The programming interface of the framework is the same for standalone Windows Form applications, ASPX web application and PHP web applications. Thus, you only need to learn it once and program it in the same way for all platforms.

This document describes how to use the Login Manager component and the login management framework. It consists of following files:

- Part A – This file.
- Part B1 – It contains the first part of a Windows Form Application sample. It can be downloaded from http://www.limnor.com/support/LoginManagerPartB1.PDF
- Part B2 – It contains the second part of a Windows Form Application sample. It can be downloaded from http://www.limnor.com/support/LoginManagerPartB2.PDF
- Part C – It contains a PHP web application sample. It can be downloaded from http://www.limnor.com/support/LoginManagerPartC.PDF
- Part D – It contains an ASPX Web Project sample. It can be downloaded from http://www.limnor.com/support/LoginManagerPartD.PDF

The files for sample projects are long but with little text. Most of contents are screenshots showing step by step programming procedures of sample projects. Most steps look similar and repetitive. For beginners, please pay attention to the highlighting in the screenshots, especially the highlighting in the Expression Editor: the result of each operation depends on which parts of an expression are highlighted.

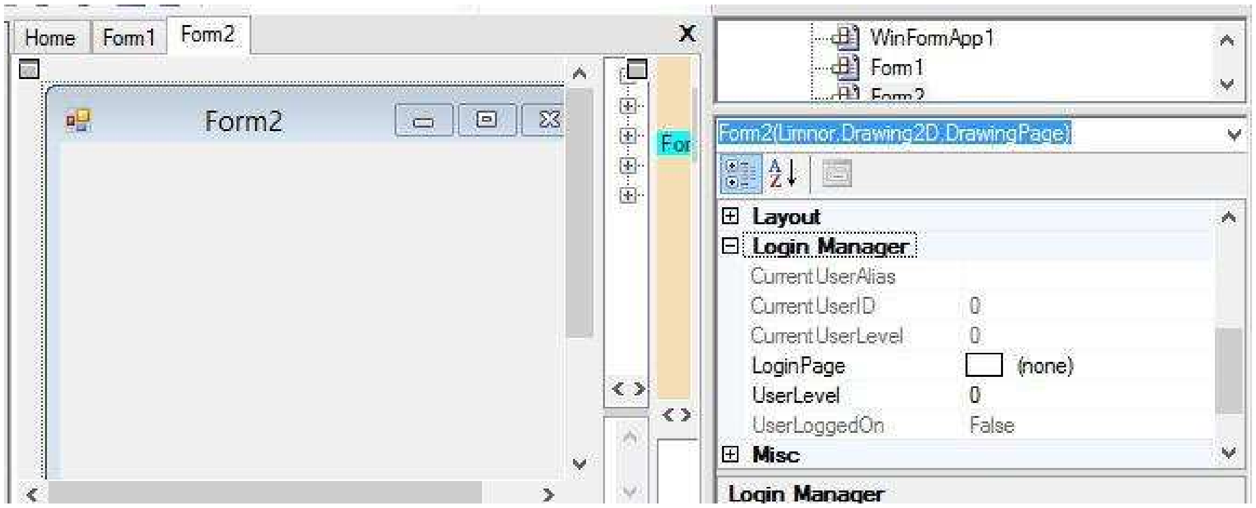
Login Management Framework

The parts of an application to be protected are forms and web pages. Login forms or login web pages should be developed to control which forms or web pages can be accessed by which users. Read this chapter through to get a basic understanding of the framework. It is OK if you do not fully understand some parts of this chapter. This chapter serves as a reference and you may go back to this chapter while reading further chapters providing sample projects.

Windows Forms

In a Windows Form application, all forms can be protected with access restrictions. A form has login management related properties, methods, and events.

Login Manage Related Properties

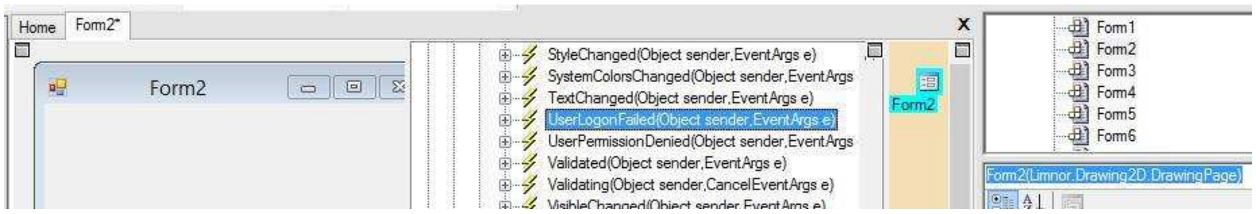


CurrentUserAlias	
CurrentUserID	0
CurrentUserLevel	0
LoginPage	(none)
UserLevel	0
UserLoggedOn	False

A Windows form has following properties for controlling access permissions:

- **CurrentUserAlias** – It is a read-only string. If a user has logged on then the value of this property is the user login name. If no user has logged on then the value of this property is an empty string or null. See description for UserAccountLoginFieldName property of Login Manager Component.
- **CurrentUserID** – It is a read-only integer. If a user has logged on then the value of this property is the user ID, usually an auto-number in a database. If no user has logged on then the value of this property is 0. You, as the author of the software, determine the availability of the user ID. If you choose not to make it available then the value of this property is always 0. See description for UserAccountIdFieldName property of Login Manager Component.
- **CurrentUserLevel** – It is a read-only integer. If a user has logged on then the value of this property is the user permission level. If no user has logged on then the value of this property is negative 1. You, as the author of the software, determine if user level is used in the system. If you choose not to use it then the value of this property is always -1. You may use your own permission control schemes with or without using the user level. We'll show such examples. See description for UserAccountLevelFieldName property of Login Manager Component.
- **LoginPage** – It indicates a login form you developed using Login Manager Component. If this property is empty then the form is not protected, that is, the form can be accessed by all users. If this property is set to a login form then at runtime whenever this form is loaded, the system checks the user permission for this form; the login form is launched automatically if needed.
- **UserLevel** – It is an integer. It indicates the user permission required by this form. The larger the value of this property, the less restriction of the form. For example, if a form's UserLevel is 3 then all users with user level (see CurrentUserLevel property) 0, 1, 2, and 3 may access the form; but a user with user level 4 cannot access the form. If this property is set to a negative value then the user level checking is disabled, that is, any logged on user may access the form.

Login Manage Related Events

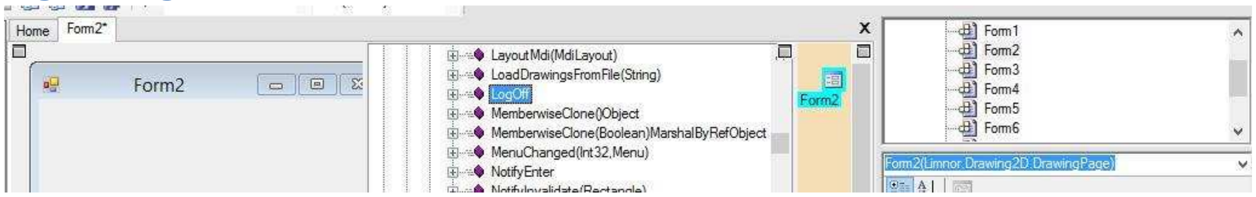


StyleChanged(Object sender,EventArgs e)
SystemColorsChanged(Object sender,EventArgs
TextChanged(Object sender,EventArgs e)
UserLogonFailed(Object sender,EventArgs e)
UserPermissionDenied(Object sender,EventArgs
Validated(Object sender,EventArgs e)
Validating(Object sender,CancelEventArgs e)
VisibleChanged(Object sender,EventArgs e)

Form2(Limnor.Drawing2D.DrawingPage)

- **UserLogonFailed** – It occurs when logon failed. When a protected form is to be displayed, if the system determines that a user log on is required then the login form will be automatically launched as a dialogue box. If the dialogue box is closed without a successful user logon then this event occurs. You may use this event to display a warning message box or other actions as required by your business. After this event, the form will be closed to prevent access to the form.
- **UserPermissionDenied** – It occurs when a logon is successful but the logged on user's level is not good for the form. Suppose the UserLevel property of the form is 2 and the logged on user's level is 3 or larger then this event occurs. You may use this event to display a warning message box or other actions as required by your business. After this event, the form will be closed to prevent access to the form.

Login Manage Related Methods



LayoutMdi(MdiLayout)
LoadDrawingsFromFile(String)
LogOff
MemberwiseClone()Object
MemberwiseClone(Boolean)MarshalByRefObject
MenuChanged(Int32,Menu)
NotifyEnter
NotifyInvalidate(Rectangle)

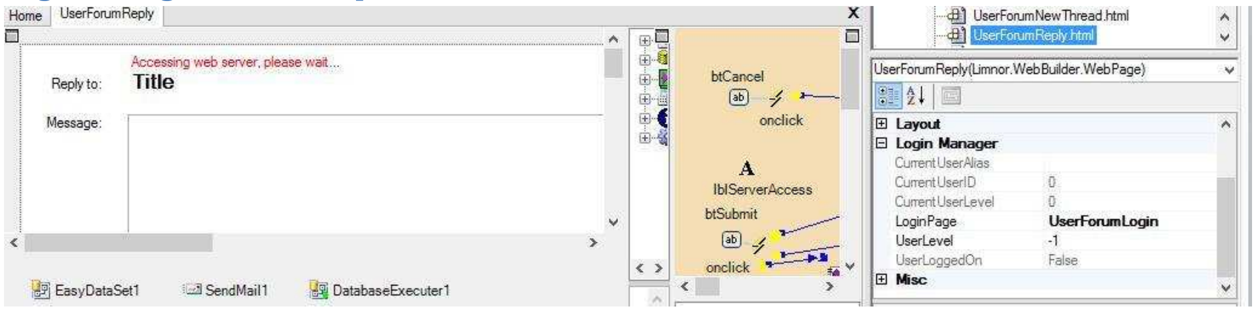
Form2(Limnor.Drawing2D.DrawingPage)

- **LogOff** – It logs off the current logged on user. Note that on executing this method, all forms with the same LoginPage property will be automatically closed to prevent access to protected forms. You should be aware of this effect.

Web Pages

In a web application (ASPX or PHP), all web pages can be protected with access restrictions. A web page has login management related properties, methods, and events.

Login Manage Related Properties



UserForumReply(Limnor.WebBuilder.WebPage)

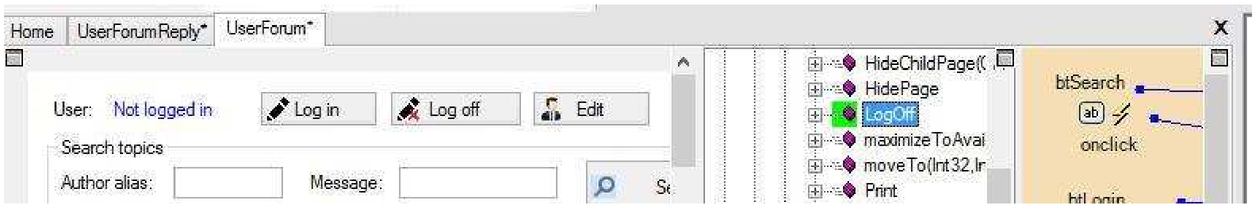
Layout	
Login Manager	
CurrentUserAlias	
CurrentUserID	0
CurrentUserLevel	0
LoginPage	UserForumLogin
UserLevel	-1
UserLoggedOn	False
Misc	

The properties related to login management framework for a web page are the same for a Windows Form, except that the LoginPage property points to a login web page, instead of a login form.

Login Manage Related Events

There is not an event on a protected web page related to login management.

Login Manage Related Methods

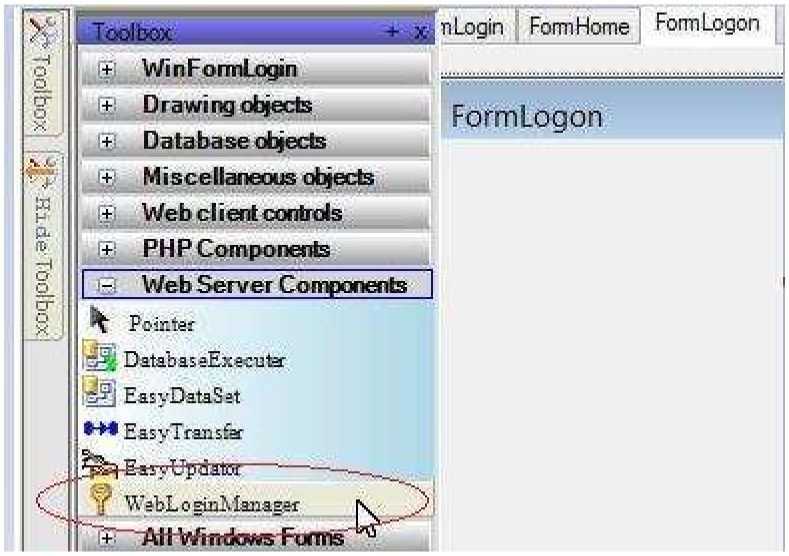


As with a Windows Form, a web page has a LogOff method. On executing this method, all web pages with the same LoginPage property will be closed automatically.

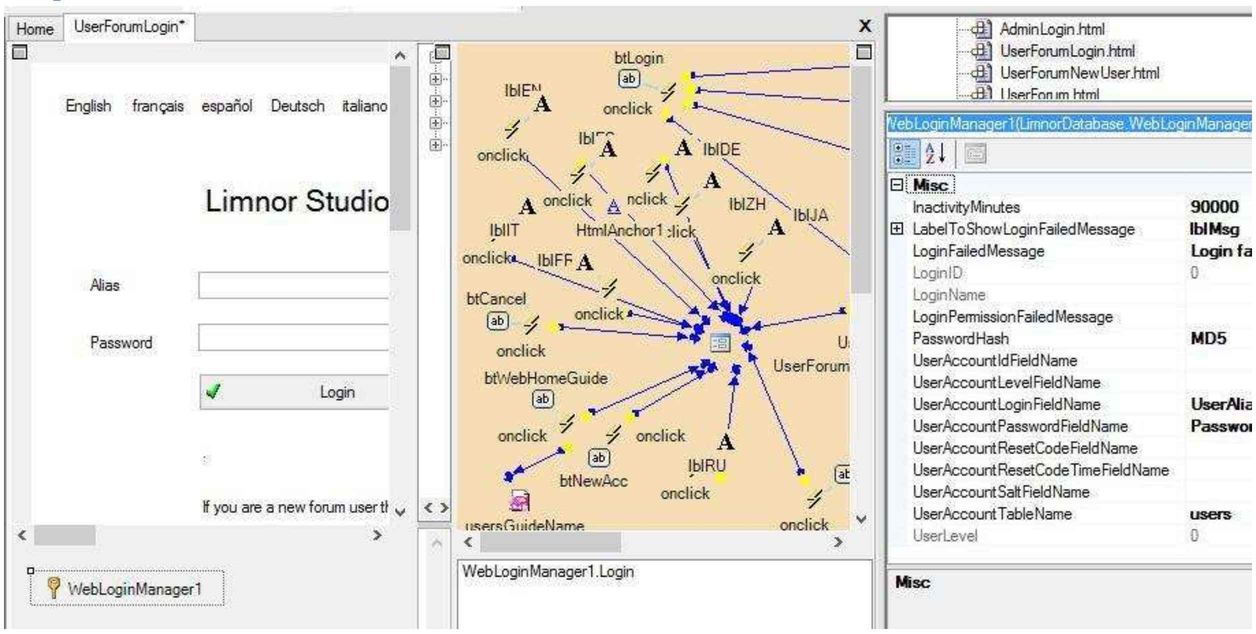
Login Manager Component

To protect a Form or a web page, we must set its LoginPage property to a login form or a login web page. A login form or web page must contain a Login Manager component.

The name of the Login Manager component is WebLoginManager. It is thus named because the login management framework was first added to web applications. The Login Manager component is listed under “Web Server Components” in the toolbox. But it can also be used in Windows Form applications.



Properties



Misc	
InactivityMinutes	90000
LabelToShowLoginFailedMessage	lblMsg
LoginFailedMessage	Login fa
LoginID	0
LoginName	
LoginPermissionFailedMessage	
PasswordHash	MD5
UserAccountIdFieldName	
UserAccountLevelFieldName	
UserAccountLoginFieldName	UserAlia
UserAccountPasswordFieldName	Passwor
UserAccountResetCodeFieldName	
UserAccountResetCodeTimeFieldName	
UserAccountSaltFieldName	
UserAccountTableName	users
UserLevel	0

Misc

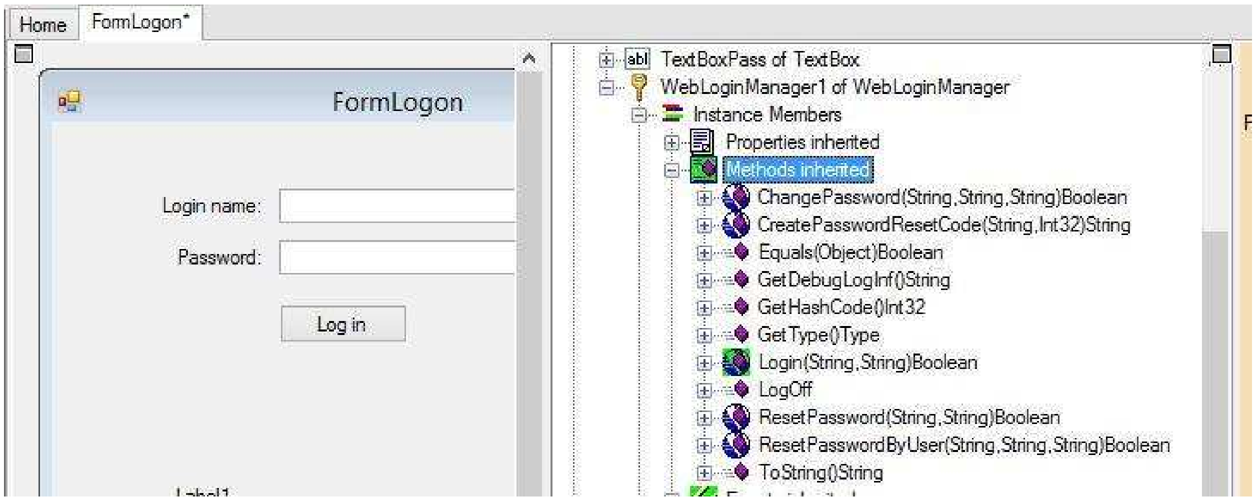
- **InactivityMinutes** – Optional. It is an integer indicating how long of inactivity of mouse and keyboard, in minutes, log in will expire. If it is 0 then this property is not used.
- **DatabaseConnection** – Required. It points to the database connection for accessing the user account table.
- **LabelToShowLoginFailedMessage** – Optional. It points to a label on the form or web page for showing login failure message.
- **LoginFailedMessage** – Optional. Message to be displayed when login failed. If this property is empty then message "Login failed" is displayed.
- **LoginID** – Read-only. On executing a succeful Login action, the value of this property is the user ID. See description of the CurrentUserID property of a form or a web page.
- **LoginName** – Read-only. On executing a succeful Login action, the value of this property is the user login name. See description of the CurrentUserAlias property of a form or a web page.
- **LoginPermissionFailedMessage** – Optional. Used only by web applications. Message to be displayed when login is successful but the user level is not good for the web page to be protected. If this property is empty then message "You do not have permission to visit this web page." is displayed.
- **PasswordHash** – Required. It indicates the hash algorithm used to generate password hash. Supported algorithms are MD5, SHA1, SHA256, SHA384, and SHA512. If you do not know which algorithm to use then choose SHA256.
- **UserAccountIdFieldName** – Optional. If the user account table uses an Auto-number then set this property to the name of the auto-number field. On logging in, the value of the CurrentUserID property of a form or a web page will be set to the value of the auto-number field.
- **UserAccountLevelFieldName** – Optional. If the user account table uses an integer to indicate user permissions then set this property to the name of the user level field. The login

management framework uses this field to control access permissions. Every form or web page has a UserLevel property to indicate the permission requirement. A user's level must be equal or less than the UserLevel property of the form or web page in order to be allowed to access the form or web page. For example, suppose a user's level is 2, then the user may access forms and web pages of UserLevel property 2, 3, or larger; but the user cannot access a form or web page of UserLevel property 1 or 0. See description for CurrentUserLevel property of a form or a web page. On successfully logging on, the value of the CurrentUserAlias property of protected forms or web pages is the value from this field.

- **UserAccountLoginFieldName** – Required. It must be set to the name of the user login name field. A user logs on by entering correct login name and password. On successfully logging on, the value of the CurrentUserAlias property of protected forms or web pages is the login name.
- **UserAccountPasswordFieldName** – Required. It must be set to the name of the user password field. Real password is not saved in the field. Only a password hash is saved. The size of this field must be large enough to hold password hash. The size required depends on the hash algorithm (in bytes): MD5:32, SHA1:40, SHA256:64, SHA384:96, SHA512:128
- **UserAccountResetCodeFieldName** – Optional. In the case of forgetting password, there should be a way to reset password, securely. The login management framework uses such a process to reset password: create a reset code and save a hash of the reset code to a field, reset-code-field, of the user account table; save an expiration date time to another field, reset-expire-field, of the user account table; send the reset code to the user (for example, via email or other secure ways); a form UI or web UI is provided for the user to enter the reset code and new password; the framework verifies the reset code and expiration time, and sets the new password. This property should be set to the name of reset-code-field. The field size should be large enough for the hash algorithm. See above description for UserAccountPasswordFieldName.
- **UserAccountResetCodeTimeFieldName** – Optional. It should be set to the name of reset-expire-field. The field type should be date-time. See description above.
- **UserAccountSaltFieldName** – Optional. To make the password-storage more secure, a common and effective approach is to add salting in password hashing. To enable this feature, add a string field of at least 64 bytes in the user account table; set the name of the field to this property.
- **UserAccountTableName** – Required. It must be set to the name of the user account table.
- **UserLevel** – Read-only. On executing a succeful Login action, the value of this property is the user level. See description of the CurrentUserLevel property of a form or a web page.

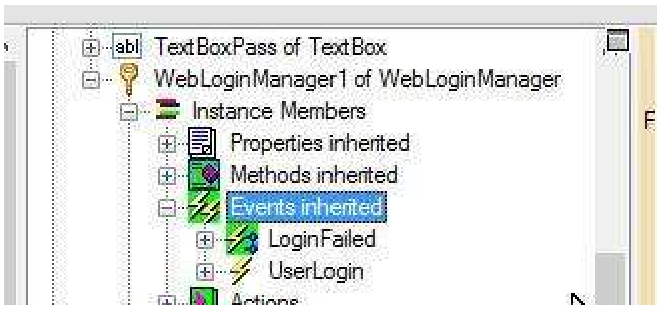
Tips: set DatabaseConnection first; once you have a valid DatabaseConnection property connecting to your database, you can set UserAccountTableName by selecting the user account table from a drop down list of table names; once you select the correct user account table name, you may set all the properties of UserAccount*FieldName by selecting from a drop down list of all field names of the user account table.

Methods


Home
FormLogon*
FormLogon
Login name:
Password:
Log in
Label1
TextBoxPass of TextBox
WebLoginManager1 of WebLoginManager
Instance Members
Properties inherited
Methods inherited
ChangePassword(String,String,String)Boolean
CreatePasswordResetCode(String,Int32)String
Equals(Object)Boolean
GetDebugLogInf()String
GetHashCode()Int32
GetType()Type
Login(String,String)Boolean
LogOff
ResetPassword(String,String)Boolean
ResetPasswordByUser(String,String,String)Boolean
ToString()String

- **Login**(loginName, password) – It uses the login name and password to log in.
- **ChangePassword**(loginName, currentPassword, newPassword)– It allows a user to change his/her password.
- **CreatePasswordResetCode**(loginName, expiratonInMinutes) – It returns a reset code for the user specified by parameter loginName. The reset code will expire from the time this method is executed in a number of minutes specified by parameter expiratonInMinutes. Usually your program should automatically send the reset code to the user, for example, via email or some other secure ways. Your program should avoid displaying the reset code and manual accessing of the reset code other than the user. For this method to work, the properties UserAccountResetCodeFieldName and UserAccountResetCodeTimeFieldName must be set to correct field names.
- **ResetPasswordByUser**(loginName, resetCode, newPassword) – It allows a user to reset his/her password. The user must get a valid reset code created by a CreatePasswordResetCode action.
- **ResetPassword**(loginName, newPassword) – It unconditionally resets a user password. A program should never directly execution this method from a UI for end users. This method is provided for creating an initial password for a new user. Your program may provide “Create New User” functionality; after creating a new user your program may use this method to set the password for the new user and send the new password to the user. The new user should immediately change the password to his/her choice.

Events


TextBoxPass of TextBox
WebLoginManager1 of WebLoginManager
Instance Members
Properties inherited
Methods inherited
Events inherited
LoginFailed
UserLogin
Actions

- **LoginFailed** – It occurs when a user tries to log on and fails. You do not have to handle this event if setting LabelToShowLoginFailedMessage property is enough to provide notification to the user.
- **UserLogin** – It occurs when a user logs on. Usually you do not need to handle this event because a protected form or web page will be automatically opened. You may handle this event to implement your own permission control approaches if user level control does not meet your requirements.

Samples Projects

Windows Forms Samples

See

http://www.limnor.com/support/LoginManagerPartB1.PDF

http://www.limnor.com/support/LoginManagerPartB2.PDF

PHP Web Project Sample

See http://www.limnor.com/support/LoginManagerPartC.PDF

ASPX Web Project Sample

See http://www.limnor.com/support/LoginManagerPartD.PDF