SecMod

User Access system of Plato

SecMod takes care of user validation in the Plato sample program.

Currently new webapps are provided with user and session tables and a login dialog. And that’s it.

SecMod should be an alternative that provides more complete set of commonplace functionality from the word “go”

Ambition:

To significantly reduce time spent implementing household user functionality for new applications

Feature overview

- Interface for desktop, mobile and windows ui’s
- “Remember me” feature
- “Forgot password” feature
- 2-factor authentication
- Self signup capable
- Uses password encryption
- Easy addition of “application parameters”
- “Zero effort” backoffice administration (system setup via windows app)
- (Embedded database or SQL backend)

SecMod tables

SecMod relies on these tables:

`SecModUser`	Replaces `WebAppUser`
`SecModSession`	Replaces `WebAppSession`
`SecModPasswordHistory`	Checking against password reuse
`SecModUserReset`	When user clicks “forgot password” it goes here
`SecModAuthToken`	When user selects “remember me” it goes here
`SecModSignupWhitelist`	If self-signup and white-listing is enabled, this is the filter
`SecModLog`	All account manipulations are registered here
`SecModSetting`	A table for Secmod settings.

Demo

- Run Plato
- Open order, report, send a mail, upload a file
- Go through SecMod UI, user admin, parameters, mail setup

oSecModConfig.pkg

A package of that name must be present in the AppSrc folder.

Much of SecMod behavior is defined there.

```
Use SecMod\cSecModConfig.pkg

Object oSecModConfig is a cSecModConfig
    Set psApplicationTitle  to "WebOrder 20"
    Set psLoginLogo_WebPath to "Images/PoweredByDataFlex.png"
    --- and a lot more ---
End_Object
```

- Show oSecModConfig.pkg for Plato

Demo

1. Run the zero-effort backoffice admin (in plato)
2. Easy parameters
    a. Secmod\oSecModSettingsManager.pkg
    b. PlatoSettingsManager.pkg

The oSessionManager

```
Object oSessionManager is a cSecModWebSessionManager // defined like this in Plato
End_Object

Session properties:
        Property String  psLoginName ""
        Property String  psUserName ""
        Property Integer piUserId 0
        Property String  psUserUUID ""

        Property Integer piUserRights 0
        Property Integer piUserRole 0
        Property Boolean pbUserAdmin False
        Property Boolean pbDataAdmin False


If (pbUserAdmin(ghoWebSessionManager)) begin
  ...
```

A SecMod dialog - example

```
Use cSecModDialogHelper_SetPW.pkg

Object oSecModPanelSetPassword is a dbModalPanel
    Set Label To "Set password"

    On_Key kEnter Send KeyAction of oButtonSetPassword

    // This makes SecMod aware that we have a “set pw” dialog. And that’s all that’s needed.
    Object oSecModHelper is a cSecModDialogHelper_SetPW
    End_Object

    Object oFormNewPassword is a Form
        Set Label to "New password:"
        Set Password_State to True
        Send RegisterSecModItemControl of oSecModHelper Self C_SECMOD_UIITEM_NEWPASSWORD
    End_Object

    Object oButtonSetPassword is a Button
        Set Label to 'Set password'
        Procedure OnClick
            Send SubmitAction of oSecModHelper
        End_Procedure
    End_Object
End_Object
```

(blue lines identical in web dialog)

Sub modules (gSecModHnd)

```
Struct tSecModHnd
    Handle hoConfig
    Handle hoHasher
    Handle hoPasswordHasher
    Handle hoDataCryptographer
    Handle hoPasswordStrengthValidator
    Handle hoMailer
    Handle hoTexter
    Handle hoSql
End_Struct
```

Apart from the usual stuff that the `ghoWebSessionManager` implements SecMod "talks" to these 8 objects:

- **`hoConfig`** ("hard" and "soft" configuration settings)
- **`hoHasher`**, **`hoPasswordHasher`** and **`hoDataCryptograher`** (well, you know)
- **`hoMailer`** (mailer object)
- **`hoTexter`** (objects for sending text messages)
- **`hoPasswordStrengthValidator`** (validation of passwords against settings)
- **`hoSqlExec`** (points to a cSqlExec object that handles SecMod tables)

```
Global_Variable tSqcModHnd gSecModHnd // Defined by SecMod packages
```

To get the mailer object use symbol **`gSecModHnd.hoMailer.`**

`gSecModHnd.hoMailer` has a SendMail method:

```
Function SendMail String sText String sSubj String sTo String sCc String sBcc Returns Boolean
```

Or one could implement an alternative "password validator" in swap that in **`hoPasswordStrengthValidator`**

Note on encryption keys

A WebApp.src has long time included a **`LoginEncryptionKey.inc`** with a password defined. This handles encryption of the database login.

With SecMod it also needs to have a **`SecModEncryptionKey.inc`** that defines a password that will be passed to the `hoDataCryptographer` methods. In other words: it will be used to encrypt user passwords.

In addition there is a third one needed for the encryption of settings stored by a `cSettingsTableManager` object. This is responsible for the encryption of smtp-server user and passwords. Its filename is **`SettingsDataEncryptionKey.inc`**.

SecMod

- Relation free. No table positions assumed (filelist.cfg)
- Table names changed. Immediate identification. No collision
- All packages are away from appsrc root
- Data dictionaries are not copied to DDSrc
- There are no additions to css or javascript
- Automatic ui for application settings
- “No code” backoffice admin program

Bonus demo

Portal configuration

Explain Portal concept

Show FVBS, Portal