---+ Package `TWiki::Users::Password`

Base class of all password handlers. Default behaviour is no passwords, so anyone can be anyone they like.

The methods of this class should be overridden by subclasses that want to implement other password handling methods.

ClassMethod new `($session) -> $object`

Constructs a new password handler of this type, referring to $session for any required TWiki services.

ObjectMethod finish `()`

Break circular references.

ObjectMethod readOnly `() -> boolean`

returns true if the password database is not currently modifyable also needs to call $this->{session}->enter_context('passwords_modifyable'); if you want to be able to use the existing TWikiUserMappingContrib ChangePassword topics

ObjectMethod fetchPass `($login) -> $passwordE`

Implements TWiki::Password

Returns encrypted password if succeeds. Returns 0 if login is invalid. Returns undef otherwise.

ObjectMethod checkPassword `($login,$passwordU) -> $boolean`

Finds if the password is valid for the given user.

Returns 1 on success, undef on failure.

ObjectMethod removeUser `($login) -> $boolean`

Delete the users entry.

ObjectMethod setPassword `($login,$newPassU,$oldPassU,$mcp) -> $boolean`

If the $oldPassU matches matches the user's password, then it will replace it with $newPassU.

If $oldPassU is not correct and not 1, will return 0.

If $oldPassU is 1, will force the change irrespective of the existing password, adding the user if necessary.

If $mcp is true, the "must change password" flag is set

Otherwise returns 1 on success, undef on failure.

encrypt($login, $passwordU, $fresh) -> $passwordE

Will return an encrypted password. Repeated calls to encrypt with the same login/passU will return the same passE.

However if the passU is changed, and subsequently changed *back* to the old login/passU pair, then the old passE is no longer valid.

If $fresh is true, then a new password not based on any pre-existing salt will be used. Set this if you are generating a completely new password.

ObjectMethod error `() -> $string`

Return any error raised by the last method call, or undef if the last method call succeeded.

ObjectMethod isManagingEmails `() -> $boolean`

Determines if this manager can store and retrieve emails. The password manager is used in preference to the user mapping manager for storing emails, on the basis that emails need to be secure, and the password database is the most secure place. If a password manager does not manage emails, then TWiki will fall back to using the user mapping manager (which by default will store emails in user topics)

The default ('none') password manager does **not** manage emails.

ObjectMethod getEmails `($login) -> @emails`

Fetch the email address(es) for the given login. Default behaviour is to return an empty list. Called by Users.pm. Only used if `isManagingEmails -> true`.

ObjectMethod setEmails `($login,@emails) -> $boolean`

Set the email address(es) for the given login name. Returns true if the emails were set successfully. Default behaviour is a nop, which will result in the user mapping manager taking over. Called by Users.pm. Only used if `isManagingEmails -> true`.

ObjectMethod findUserByEmail `($email) -> \@users`

Returns an array of login names that relate to a email address. Defaut behaviour is a nop, which will result in the user mapping manager being asked for its opinion. If subclass implementations return a value for this, then the user mapping manager will **not** be asked. Only used if `isManagingEmails -> true`.

Called by Users.pm.

ObjectMethod getMustChangePassword `($cUID) -> $flag`

Returns 1 if the $cUID must change the password, else 0. Returns undef if $cUID not found.

ObjectMethod getUserData `($cUID) -> $dataRef`

Return a reference to an array of hashes with user data, used to manage users. Each item is a hash with:

- `{name}` - name of field, such as "email"
- `{title}` - title of field, such as "E-mail"
- `{value}` - value of field, such as "jimmy@example.com"
- `{type}` - type of field: `text`, `password`, `checkbox`, `label`
- `{size}` - size of field, such as `40`
- `{note}` - comment note, if any

User management forms can be build dynamically from this data structure. Each password manager may return a different set of fields.

ObjectMethod setUserData `($cUID,$dataRef)`

Set the user data of a user. Same array of hashes as getUserData is assumed, although only `{name}` and `{value}` are used.

Sub classes should return an empty string if save action is OK, or an error string starting with 'Error: '.

ObjectMethod canFetchUsers `() -> boolean`

returns true if the fetchUsers method is implemented and can return an iterator of users. returns undef / nothing in this case, as we are unable to generate a list of users

ObjectMethod fetchUsers `() -> newTWiki::ListIterator(\@users)`

returns a TWikiIterator of loginnames from the password source. If AllowLoginNames is false this is used to remove the need for a TWikiUsers topic.

This topic: TWiki > TWikiUsersPasswordDotPm
Topic revision: r6 - 2011-08-21 - TWikiContributor

Note: Please contribute updates to this topic on TWiki.org at TWiki:TWiki.TWikiUsersPasswordDotPm.